

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problems Mailbox.**

Requested Patent: JP5134830A

Title:

INDICATING RESOURCE UTILIZATION IN A DATA PROCESSING  
SYSTEM. ;

Abstracted Patent: EP0518574 ;

Publication Date: 1992-12-16 ;

Inventor(s):

EMRICK SAMUEL LEE (US); HOLCK TIMOTHY MANFRED (US);  
SUMMERS JAMES HOYET (US); DEWITT JIMMIE EARL (US) ;

Applicant(s): IBM (US) ;

Application Number: EP19920305195 19920605 ;

Priority Number(s): US19910713484 19910610 ;

IPC Classification: G06F11/32 ;

Equivalents: JP2038110C, JP7074984B

ABSTRACT:

A graphical system resource monitor is provided to depict, in real-time, a data processing system's internal resource utilization. A window or viewport of a data processing system displays user specified internal system resources, such as memory, CPU, or peripheral device availability/utilization. This graphical representation of the 'state' of the data processing system's resources is maintained in real-time, while the data processing system's resources is maintained in real-time, while the impact on the system's performance in providing such information is kept to a minimum. This is accomplished through a combination of various techniques, including specialized device drivers for the respective devices coupled with a unique data reduction technique. The graphical results of these resource monitors are continually updated in real-time. This real-time support provides an immediate and accurate representation of the internal operations of the data processing system. Further, these resources can monitored at the process level of a multiprocessing system. These representations can be used by a user to identify, isolate, and fine-tune the data processing system's resources to improve the overall efficiency of the system being monitored.

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平5-134830

(43) 公開日 平成5年(1993)6月1日

(51) IntCl. <sup>5</sup>	識別記号	序内整理番号	F I	技術表示箇所
G 0 6 F 3/14	3 2 0 A	7165-5B		
9/00	3 2 0 C	7927-5B		

審査請求 有 請求項の数15(全 32 頁)

(21) 出願番号 特願平4-104357

(22) 出願日 平成4年(1992)4月23日

(31) 優先権主張番号 7 1 3 4 8 4

(32) 優先日 1991年6月10日

(33) 優先権主張国 米国 (US)

(71) 出願人 390009531

インターナショナル・ビジネス・マシーンズ・コーポレーション

INTERNATIONAL BUSINESS MACHINES CORPORATION

アメリカ合衆国10504、ニューヨーク州アーモンク (番地なし)

(72) 発明者 ジミー・アール・デユウイット

アメリカ合衆国テキサス州、ジョージタウン、サウス・メイン・ストリート 1902番地

(74) 代理人 弁理士 頓宮 孝一 (外4名)

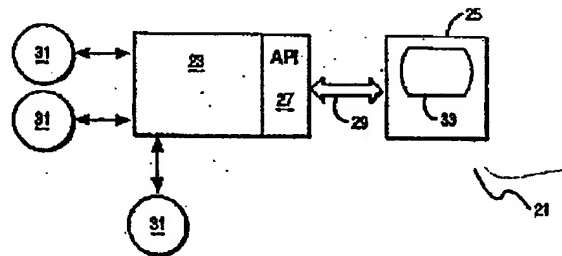
最終頁に続く

(54) 【発明の名称】 データ処理システムにおけるリアル・タイム・システム源のモニタ方法とモニタ装置

(57) 【要約】 (修正有)

【目的】 グラフィック・システム資源モニタは、データ処理システムの内部資源利用率をリアルタイムで提示。

【構成】 データ処理システムのウィンドウ又はビューポートは内部システム資源を表示する。モニタ・システム21がデータ収集機構23と資源モニタ25に概念的に分割される。アプリケーション・プログラミング・インタフェースAPI27が、これらの2つのオペレーティング・モデル間のインタフェースとして使用される。パイプ29は機構23と資源モニタ25間の接続に使用。データ収集機構23はトレースされる様々な資源31のために重要な性能データを収集する。これらの資源モニタのグラフィカルな結果は、リアルタイムで絶えず更新。これらの資源は多重処理システムのプロセスレベルでモニタ出来る。これらの表示は、モニタされるシステムの総合効率を改善するためにデータ処理システムの資源の確認、分離、微調整用にユーザにより使用できる。



1

2

## 【特許請求の範囲】

【請求項1】データ処理システムによって上記データ処理システムの少なくとも1つのプロセスをモニタリングするステップと、

上記モニタリングからの結果としての上記少なくとも1つのプロセスに対して資源使用データを生成するステップと、

上記データ処理システムの上記資源使用データをリアル・タイムで表示するステップとを有するデータ処理システムの資源利用率を示す方法。

【請求項2】上記少なくとも1つの資源が、ランダム・アクセス・メモリであることを特徴とする請求項1記載の方法。

【請求項3】上記少なくとも1つの資源が、周辺装置デバイスであることを特徴とする請求項1記載の方法。

【請求項4】上記周辺装置デバイスが、何れの直接アクセス記憶装置デバイス又は交信アダプタであることを特徴とする請求項3記載の方法。

【請求項5】リアル・タイムで表示される上記ステップの上記資源使用データが、データ処理システムのディスプレイに表示されることを特徴とする請求項1記載の方法。

【請求項6】上記資源使用データが、上記データ処理システムのディスプレイのウィンドウに表示されることを特徴とする請求項5記載の方法。

【請求項7】上記資源使用データが、ローカル・データ処理システムのディスプレイに表示されることを特徴とする請求項5記載の方法。

【請求項8】上記資源使用データが、遠隔のデータ処理システムのディスプレイに表示されることを特徴とする請求項5記載の方法。

【請求項9】上記モニタリングするステップが、デバイス・ドライバによって部分的に実行されることを特徴とする請求項1記載の方法。

【請求項10】上記モニタリングするステップが、制御プログラムと共に協調してデバイス・ドライバによって実質的に実行されることを特徴とする請求項1記載の方法。

【請求項11】データ多重処理システムによる上記データ処理システムによって少なくとも1つのプロセスをモニタリングするためのモニタ手段と、

上記モニタリングからの結果としての上記少なくとも1つのプロセスに対して資源使用データを生成するための生成手段と、

上記データ処理システムの上記資源使用データをリアル・タイムで表示するための表示手段とを有するデータ処理システムの資源利用率を表すためのシステム。

【請求項12】上記少なくとも1つの資源が、ランダム・アクセス・メモリであることを特徴とする請求項11記載のシステム。

【請求項13】上記少なくとも1つの資源が、周辺装置デバイスであることを特徴とする請求項11記載のシステム。

【請求項14】上記データ処理システムによって上記データ処理システムのランダム・アクセス・メモリのワーキング・セットをモニタリングするステップと、

上記モニタリングからの結果としてのランダム・アクセス・メモリのワーキング・セットのために資源使用データを生成するステップと、

10 上記データ処理システムの上記資源使用データをリアル・タイムで表示するステップとを有するデータ処理システムの資源利用率を表すための方法。

【請求項15】データ処理システムの資源利用率を表す手段を有し、コンピュータ可読の媒体に常駐するコンピュータ・プログラム・プロダクトであって、

上記データ多重処理システムによって上記データ処理システムの少なくとも1つのプロセスをモニタリングするためのモニタ手段と、

20 上記モニタリングからの結果としての上記少なくとも1つのプロセスのために資源使用データを生成するための生成手段と、

上記データ処理システムの上記資源使用データをリアル・タイムで表示するためのディスプレイ手段とを有するコンピュータ・プログラム・プロダクト。

## 【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、データ処理システムに関し、特に、データ処理システムの資源利用のグラフィカルなモニタに関する。

30 【0002】

【従来の技術】データ処理システムの効率及び活用のために追及され続けられた改善において、様々な種類のデータ・モニタが、これらのシステム内で何が生じているのか理解するために、ユーザの手助けとして開発された。データ処理システムの重要な資源には、ランダム・アクセス・メモリ(RAM)の使用率、周辺装置デバイスの使用率および中央演算処理装置(CPU)のビジー/アイドル時間のような物が含まれる。これらの資源は、データ処理システム全体の処理能力の効率を高めるためにデータ処理システムのオペレータに対して様々なシステム・パラメータの微妙な調整における重要な情報を与える。

40 【0003】オペレーティング・システムのユーザは、どのくらいのメモリが使用されるかの情報を必要とする。メモリ利用の情報としては、特に、メモリ・ワーキング・セットが有用で、現在使用するアプリケーションに対して、コンピュータの物理メモリの余裕が十分であるかどうかを示す。不十分なメモリ割り付けは、このメモリ不足に基づいて起こる、過大なスワッピングまたは  
50 ページングのために能率の悪いシステム・オペレーショ

3

ンを生む。従来のプロダクトの解析システム・メモリは、システムに存在するシステム・メモリの実容量にもよるが、実行するのに一般に約15〜45秒程かかる。提示情報は、個々のアプリケーションによるRAMの消費量を求めるのに有用であるが、モニタされるシステムにツールを挿入するだけで、要素ではない。これらの従来のプロダクトは、テキスト・スクリーンを使用する。他の従来の技術及びツールは、RAM使用を測定、或いは計算するために特殊化されたハードウェア援助に頼った。

【0004】様々な技術が、様々なタイプのデータ処理システム資源を測定するのに使用されている。システム自身によって直接に内部をモニタする技術は既存する技術の1つである。これらの技術は、データを捕らえ、ある種の大規模な記憶装置に書込むのに、一般に、データ処理システム自身の相当な資源を消費する。次に、後続する手順が、このデータを読み出し、分析するために使用する（すなわち、分析はリアル・タイムではない）。

【0005】周辺装置デバイスのデバイス利用率は、各I/O（入出力）の開始及び終了時間を正確に測定することで、以前から計られていた。これは、個々のI/O時間の計算になる。所定の期間でこのI/O時間を合計することで全体のビジー・タイムを計算することが可能であった。次に、デバイス利用率が、全体のビジー時間を合計経過時間で割ることによって計算される。このアプローチには、2つの問題がある。第1に、I/O（普通、デバイス・ハードウェア及び又はオペレーティング・システム）測定とI/O開始及び停止時間の記録を直接に管理するエンティティが必要なことである。第2に、これらのI/O事象を正確に計る精度の高いハードウェア・タイマーを必要とする。あるシステム、例えば、パーソナル・コンピュータはこれらの基準に合致していない。換言すると、ハードウェアまたはオペレーティング・システムはI/O時間を測定しない。さらに、ハードウェア・タイマーは、今日のパーソナル・コンピュータのほとんどは、I/Oタイミングを正確に測定できないほど精度が低い（32ミリ秒）。このように、既存のパーソナル・コンピュータ・システムでのデバイス利用率は、これらの従来の方法を用いて得ることはできない。

【0006】データ処理システムのCPUアイドル・タイムは、コンピュータの中央演算処理装置（CPU）が何れのタスクによっても利用されていない時間の量である。従来のCPUアイドル・タイムの測定は、スレッドを用いて一連のタスクを実行した。スレッドが実行されたタスク数は、次に、そのスレッドがすべての利用可能なCPUタイムで実行されたならば、達成されたであろうタスクの仮説的な数と比較される。このプロシージャは、タスクの仮説的な数が、異なるデータ処理システムでは異なるという点において欠陥がある。タスクの実行

4

に必要な最小時間を求めるのにシステムの特定の較正アルゴリズムが必要とされる。この較正方法は、システム間で作動するときに、信頼性がなく、多くの実際の問題を提起する。

【0007】

【発明が解決しようとする課題】一般に、前述のタイプのシステムは、性能データが収集され、さらに分析するためにギャザリング・システムによって比較的遅い大量記憶装置に書き込まれる点で、又、欠陥を有する。これは、データ分析の方法よりも速くオペレーティングするデータを捕える方法に起因する。このように、大量記憶装置は、方法が異なる動作速度で働くように、バッファとして使用される。さらに、データ・ギャザリング・システムによって生成されたデータは、分析方法が大量のデータを管理、又は維持できないほどの大容量の性質を有する。この制約で、さらに中間の大きさの記憶装置を必要とする。

【0008】この中間のバッファリングのために分析は、リアル・タイムで実行できないばかりか遅れる。このように、システム性能及び動作の何れのレポートまたは他のタイプのフィードバックは、実際の性能では常に時代遅れである。今日のデータ処理システムは、多数のタスク及びユーザをサポートする等、より複雑な動作環境をサポートしており、遂行能力に関するデータの遅延は、システムの運営に重大な障害となり、何れの問題が発生しても、検知する前に障害が生じるような報告無しと同じ状態となる。

【0009】データの分析に使用される他の方法は、莫大な量のCPUのような、ギャザリング・システムの資源を必要とする。この結果、分析は、リアル・タイムで実行されず、資源を大きな比率で消費することとなり、データを歪めて、根本的なシステム・オペレーションとしての意義がなくなる。

【0010】あるシステムが、上述の限界を克服するのを試みたが、しかし、多重処理システムの処理段階で情報を保持し、又は捕えるのに失敗した。むしろ、システムの性能を劣化させる原因となる特定の処理を目的とする能力のない、全体的なシステムの使用が観測された。このプロセス解明の失敗は、全体的なシステムは性能が悪く、システムのどのプロセスが原因であるかの指摘は、無意味なことを示している。

【0011】

【課題を解決するための手段】本発明は、メモリ、CPUまたはリアル・タイムでの周辺装置デバイスの可用性と利用性のような、ユーザ指定の内部システム資源をグラフィック・ディスプレイで表すことによって前述の問題と欠陥を解決する。データ処理システムの資源状態のグラフィカルな表現は、このような情報の提供によるシステム性能への影響が最小限度に保たれる一方で、リアル・タイムで管理される。これは、ユニークなデータ

整理技術と共に、特殊化されたデバイス・ドライバを含む様々な技術の組合せによって達成される。これにより情報が多重処理システムの細分化された処理で得られ、所定の処理のための資源が、モニタできる。これらの資源モニタのグラフィカルな結果は、データ処理システムのディスプレイのウィンドウ又はビューポートでユーザに便利よく提示され、及びデータは、リアル・タイムで更新される。リアル・タイム・サポートは、データ処理システムのオペレーションを直接に及び正確な表示を提供する。これらの表示は、モニタされるシステムの総合効率を改善するためにデータ処理システムを確認、分離、及び微調整するためにユーザによって使用できる。

【0012】RAMワーキング・セットを含むメモリ利用には、デバイス・ドライバで、最後に使用された(LRU) (Least Recently Used) タイムスタンプを比較するために非常に効率の良いメカニズムが使われる。

【0013】ディスク駆動装置又は通信ラインを始めとする周辺装置デバイス利用率測定のために、周辺装置デバイスがビジーである時間の平均値が計算される。この測定に使用される方法では、追加のデバイス・ファームウェア援助、又はデバイス利用率情報を得る、従来の方法で必要であった精度の高いタイミング・ハードウェアは、必要とされない。最小オペレーティング・システム援助は、周辺装置のデバイス・ドライバでフックを使用して成し遂げられる。

【0014】CPU利用率においては、プロセスは、処理システムのレベルで最も低い優先権、すなわち、アイドルが割り当てられる。オペレーションのアイドル事象の時間量が、CPUアイドル・タイムを表す。

【0015】本発明の目的は、改善された資源モニタを提供することである。

【0016】本発明の目的は、リアル・タイムで働く資源モニタを提供することである。

【0017】本発明の目的は、リアル・タイムで働く内蔵データ処理システム・モニタを提供することである。

【0018】本発明の目的は、リアル・タイムで働く内蔵データ処理システム・プロセス・モニタを提供することである。

【0019】

【実施例】全体について概説する。次の方法とシステムは、RAM利用率、CPUアイドル・タイム、そして周辺デバイス利用率を含むデータ処理システムの資源をモニタするユニークな方法を説明する。このモニタリングは、システムの内部、又は従来の通信方法を経て、付属の遠隔のデバイスの何れにも実行できる。様々なモニタリング及びトレース技術が、モニタされる各々の資源に利用される。データは、多重処理環境のプロセス・レベルで捉え、提示することができる。データ・キャッシュのような他の種類のデータ処理システム資源が、本発明の請求範囲内で、本発明と同様な技術を利用して同様に

モニタできる。全体的スキーマは、モニタされた変数のユーザによる修正のために、資源パラメータ及びサポートのリアル・タイムのグラフィカルな操作が簡単なシステムに必須として実装される。以下の説明のために、リアル・タイムは、ウェブスターの新カレッジ辞典(Webster's New Collegiate Dictionary) によって次のように定義されていることに注意されたい。リアルタイムは、その発生が実際に同時である事象の報告又は記録で事象が生じる実際の時間を意味する。

【0020】ここで図1を参照するに、開示されたモニタ・システム21が、2つの独立したオペレーションの、データ収集機構(DCF)23及び資源モニタ(RM)25に概念的に分割されている。アプリケーション・プログラミング・インタフェース27、すなわち、APIが、これらの2つのオペレーティング・モデル間のインタフェースとして使用される。名前付きパイプ29は、当業者には既知であり、及び"IBM OS/2 プログラミング・ツール及び情報、バージョン1.2" (IBM OS/2 Programming Tools and Information, Version 1.2) で詳細に説明されているので参照されたい。ここでは、バックグラウンド材料として、DCF23とRM25間の接続に使用されている。DCF23は、トレースされる様々な資源31のために重要な性能データを収集する。RM25は、資源使用の描写を表す。好ましい実施例では、この資源使用の描写は、従来のデータ処理システム・ディスプレイ33に図表で表示される。名前付きパイプ29を使用し、システムが従来の通信技術で接続されると、図示されているシステムは、資源モニタ25を実行させているコンピュータとは別のコンピュータで動いているデータ収集機構23に接続できる。これは、名前付きパイプを使用することにより、ネットワーク透過オペレーションが可能になるからである。

【0021】ここで図2を参照するに、システム21は、データ収集技術35、データ整理技術37及び提示技術39に、機能的に3つのオペレーションに副分類され分割される。データ収集技術35は、周辺装置デバイスのRAMワーキング・セット利用及びサンプリングを含む。データ整理技術37は、CPUアイドル・タイムを決定するためのアイドル・スレッドの測定、トレース・データのフィルタリング及び他の整理方法を含む。最後に、提示技術39は、ダイナミックなモニタリングおよびマルチ・ビューポート・ウィンドウ操作を含む。図2は、図1の概念モデルにオーバーレイされる機能的な表示をさらに例示する。図2で分かるように、データ収集技術35は、データ収集機構23内に完全に含まれる。データ提示技術39は、資源モニタ25内に完全に含まれる。データ整理技術37は、データ収集機構23及び資源モニタ25の両方に共存する。後で示されるように、整理技術37における責任の共有は、データの捉え及びモニタされる資源の高性能のグラフィカルな描写

の両方において効率性を与える。好ましい実施例でのモニタされる各資源に対する特定の的方法論については後述する。

【0022】RAM利用率について説明する。モニタできる図1のシステム資源31の1つは、RAM利用率である。開示されたモニタリング方法は、全オペレーティング・システムでのメモリ利用率を高速で計算する(ミリ秒で)。その結果をリアル・タイムで図表で表示する。好ましい実施例では、そのオペレーティング・システムは、IBM OS/2 (IBMの商標) オペレーティング・システムであるが、しかし、これらの概念は、この分野の専門家によって他のタイプのコンピュータ・オペレーティング・システムにも容易に導入できる。

【0023】ここで図3を参照するに、全物理メモリ41の様々なカテゴリが定義され、各それぞれのカテゴリの利用に比例して図表で表されている。固定メモリ43は、スワップ・アウト又は放棄できない分割化されたスワッピング・メモリ構成のメモリである。この固定メモリは、このメモリを所有するアプリケーションがロードされている限り、RAMの中に割り当てられて残る。ワーキング・セット・メモリ45は、次のように定義できる。(1) スワップ可能でもなければ、放棄可能でもない全メモリ・セグメント。(11) スワップ可能及び放棄可能で、適用できるシナリオの実行中に使用される全メモリ・セグメント。使用メモリ47は、システムによって割り当てられたRAMであり、物理メモリ内に在る(すなわち、割り当てられ及びスワップ・アウトされないメモリ)。ワーキング・セット・メモリは、瞬時値でない。このメモリは、“ワーキング・セット期間”と呼ばれる時間の間、使用されるメモリである。ワーキング・セット期間は、ダイナミック・モニタリング・セクションで説明されるように動的に変えることができる。

【0024】システム全体のワーキング・セット・メモリ45を計算するために、改善されたデバイス・ドライバが、メモリ利用の非常に速い計算を提供する。それは、ユーザによって動的に指定されるワーキング・セット期間を使用する。デバイス・ドライバは、アセンブリ言語でコード化され、性能を高めるためにリング0で実行し、プロテクトされた資源に対して制約されずにアクセスでき、従来技術によって以前に実行されたセッションによってではなく、全システムのためにワーキング・セットを得ることができる。リング0は、当業者には周知のリングで、オペレーティング・システムのコア・レベルで実行する、すなわち、CPUハードウェアに最も近いリングである。他のレベル、例えば、好ましい実施例であるOS/2のオペレーティング・システムのレベル1~3は、CPUの内部資源のより低い各々のアクセス・レベルで実行する。

【0025】図4を参照するに、示されているワーキング・セット・メモリは、最後の“ワーキング・セット期

間”秒の51でのアクセスされたメモリの百分率である。ワーキング・セット期間は、“サンプリング期間”53の時間毎に更新、又はスナップショットされる。本発明は、ワーキング・セット・メモリを計算するためにメモリ使用のスライディング・ウィンドウ55を使用する。サンプリング期間毎に取られるメモリのスナップショットは、全メモリ・セグメントの最後に使用されたLRUタイムスタンプを調べる(タイムスタンプの方法に関してはデバイス・ドライバ・セクションの中で後述される)。LRUタイムスタンプは、メモリ・セグメントがどれくらい最近アクセスされたかを示す。各スナップショット毎に、2つの値が示される。

【0026】1. メモリの全内容に対しての最新のLRUタイムスタンプ。この値は、メモリの一部分が最も最後にアクセスされた最後の時間である。この値が、“ワーキング・セット期間”秒として後で使用される。

【0027】2. どのセグメントが、“ワーキング・セット期間”秒前にセーブされたLRUタイムスタンプの値以後、アクセスされたかを示す。これらのセグメントのサイズの和が、その時間のワーキング・セット25を構成する。

【0028】手順は、次のように機能する。デバイス・ドライバは、すべての物理メモリ・ブロックを通る。各スワップ可能、放棄可能のブロックで、2つの比較を行う。

【0029】1. ブロックのLRUタイムスタンプを、ワーキング・セット期間秒前に得たLRUタイムスタンプと比較する。— ブロックのタイムスタンプが、ワーキング・セット期間のタイムスタンプよりも大きい場合、そのブロックは、ワーキング・セットにあり、ブロックサイズが、ワーキング・セット和に加えられる。

【0030】2. ブロックのLRUタイムスタンプをこれまで見つけた最大のタイムスタンプ(最新)と比較して、大きいならば、その値を現在の最大のタイムスタンプに使う。

【0031】デバイス・ドライバは、ワーキング・セットの全ブロックのサイズ(バイトで)の和および全物理メモリである最大(最新)LRUタイムスタンプを復帰する。

【0032】この手順は、図5でさらに詳細に説明されている。様々な変数が、60で初期化される。メモリの次のブロックが、72でデバイス・ドライバによって読出される。読出されたメモリのブロックサイズが、74において物理メモリのカウンタを含む変数70に加えられる。76ではブロックがフリーか、すなわち、未使用か、判定される。フリーでない場合は78でブロックサイズは、使用中メモリのカウンタを含む変数66に加えられる。次に、80でブロックがスワップ可能か、放棄可能か、判定される。否定ならば、ブロックサイズは、82で固定メモリのカウンタを含む変数68に加えられる。

る。さらに、固定メモリはワーキング・セットの一部として定義されているので、ブロックサイズは、また、86で、ワーキング・セット・メモリのカウンタを含む変数64に加えられる。ブロックがスワップ可能及び放棄可能であるならば、ブロックのLRUタイムスタンプのチェックが行われる84へ処理は続く。ブロックLRUタイムスタンプが最大ワーキング・セット期間タイムスタンプより大きいならば、ブロックサイズは、86で、ワーキング・セット・メモリのカウンタを含む変数64に加えられる。何れの場合でも、ブロックのLRUタイムスタンプが最大タイムスタンプ62より大きいかどうかの次の決定が、88で行われる。大きいならば、ブロックLRUタイムスタンプは、90で、新最大タイムスタンプ62としてセーブされる。最後に、さらにブロックが存在するかどうか、92でチェックが行われる。もしそうならば、処理は72へと続く。否定ならば、デバイス・ドライバは、94に復帰する。ここでの最大タイムスタンプ、ワーキング・セット・メモリ、使用中メモリ、固定メモリ、および物理メモリの各値は、図3で定義された値である。

【0033】後で説明されるグラフィックス・プログラムは、一定の間隔でデバイス・ドライバを呼び出し、ユーザ・マシンの全物理RAMの大きさに応じてワーキング・セット・メモリをプロットする。この呼出しは、デバイスへのシステムAPIのコールを通して交信するデータ収集機構によって実行される。情報は、図1の名前付きパイプ29を通してAPIを経てグラフィックス・プログラムに渡される。図3を再び参照するに、デバイス・ドライバは、ワーキング・セット・メモリ45を再計算するためにサンプリング期間53の時間毎に呼び出される（ワーキング・セットが物理メモリを越えると、100%として表示される）。図9でわかるように、典型的サンプリング期間は5秒で、典型的ワーキング・セ

ット期間は、60秒である。

【0034】ここで図6を参照するに、固定及び使用中メモリが、ワーキング・セット・メモリの、上部の100、下部の102のパウンドとして、それぞれグラフ化されている。ワーキング・セット・メモリ104は、固定メモリを下回る又は使用中メモリを上回することは決してないので、このグラフはワーキング・セットメモリが可能な範囲にあることを示す。この機能は、絶対的に可能な、最小及び最大限量が自動的にグラフとして表される自己補正メカニズムを提供し、ユーザを支援する。最小絶対値は、固定メモリの計算値であり、最大絶対値は、使用中メモリの計算値である。

【0035】安定したシナリオとしては、ワーキング・セット期間が短くなると、報告されたワーキング・セットは、より少ないメモリが、より短い時間で典型的にアクセスされるので、より低くなる。ワーキング・セット期間が増加すると、さらに多くのアプリケーションに対してより多くのメモリがより長い時間で典型的にアクセスされるのでワーキング・セットは長くなる。

20 【0036】ワーキング・セット期間パラメータの値は、報告されたワーキング・セット・メモリに影響を及ぼすことができる。期間を長くすると、ワーキング・セット・メモリが使用中メモリに、すなわち、上限に接近する。期間を短くすると、ワーキング・セット・メモリが固定メモリに、すなわち、下限に接近する。固定及び使用中メモリは、瞬間値である。しかしながら、ワーキング・セットは、一定の時間内での使用されたメモリとして定義される。

【0037】以下のテーブル1は、RAMモニタがどのように、システム資源を解釈するために使用できるかを述べる。

【0038】

テーブル1 RAMモニタ・シナリオ解釈

#### シナリオ 解釈

注意： ワーキング・セット期間は、すべてのシナリオで60秒に設定される。

大容量のアプリケーションがロードされる。ワーキング・セットは、大幅な増加を示す。固定メモリは、小幅な増加を示す。ユーザは、暫くの間そのアプリケーションを使用しないことにする。数分後にワーキング・セットは、ダウンする。

ロードされたプログラムは、ワーキング・セットの一部として報告され、その固定メモリは、システム固定メモリの一部として報告される（また、ワーキング・セットに含まれる）。60秒間、プログラムをロードするのに使用したメモリは、ワーキング・セットで報告され続ける。

アプリケーションは、60秒間作動しないので（従って、ほとんどのアプリケーションのメモリは、アクセスされない）、プログラムがそれでもロードされて

11

12

もワーキング・セットは、数分後、ダウンする。しかしながら、アプリケーションの固定メモリは、それでもそのワーキング・セットの一部として、及び固定メモリの一部として報告される。

大容量のアプリケーションが、ロードされる。ワーキング・セットは、予想よりも大幅な増加を示す。

このアプリケーションは、通常の実行よりも、より多くのメモリを使用する。報告されたワーキング・セットは、正常動作中にドロップする。

大容量のアプリケーションがロードされるが、しかし、直ちに終了する。報告されたワーキング・セットは、上昇後、急速に低下する。

OS/2がアプリケーションをアンロードする場合、OS/2は、アプリケーションのメモリをフリーにする。フリーにされたメモリは、ワーキング・セットに報告されない。

スワップ・イン及びスワップ・アウトのグラフは、ワーキング・セットが100%でなくても、かなりのアクティビティを示す。

新セグメントをスワップ・イン又はロードしなければならない場合、最近アクセスされなかった古いセグメントは、スワップ・アウト又は放棄する必要がある。スワップ・アウトされるメモリは、そのメモリが60秒前の以前にアクセスされたものであれば、ワーキング・セットに報告される。

時折のスワップ・アクティビティがあっても、遂行能力を良くするために十分なメモリが存在する。それ以上の物理メモリは、必然的に必要とされない。

OS/2システム及びSPMアプリケーションが最初に開始するときは、固定メモリは、期待された値よりも高い。

固定メモリは、CONFIG.SYSファイルで定義された大容量のVDISKまたはDISKCACHEを含むことができる。

安定したシナリオのために、ワーキング・セット期間は、60秒から10秒に変えられる。報告されたワーキング・セットは現在、低い。

そのワーキング・セットが低いのは、より少ないメモリが典型的に、60秒よりも10秒でアクセスされるからである。

安定したシナリオのために、ワーキング・セット期間は、60秒から1000秒に変わる。報告されたワーキング・セットは、現在、高い。

そのワーキング・セットが高いのは、より多くのアプリケーションの、より多くのメモリが、60秒とは違い、1000秒でアクセスされるからである。

【0039】メモリ利用率、特に、ワーキング・セット・メモリの情報は、コンピュータの物理メモリが、現在のアクティブなアプリケーションに対して十分であるかどうかを示すために有用である。この技術によってユーザは、“こうしたら、どうなるのか?”の疑問を、変数または当該のエンティティに影響を及ぼすパラメータを実際にリセットすることなしに、問うことができる。要約すると、この技術は、RAMのワーキング・セット、固定及び使用中メモリ量を含む、全体としてのオペレーテ

ィング・システムのランダム・アクセス・メモリ(RAM: Random Access Memory)の利用率を高速に計算し、及び図表でこれらの結果を表示する。

【0040】ダイナミックなモニタリングについて説明する。データ処理システムのユーザが、ディスプレイ・スクリーンのダイナミックなモニタのディスプレイに影響を及ぼすパラメータを変える手順について述べる。この手順は、機能が少くとも1つの変数によって影響を及ぼされる場合の、データ処理システム・ディスプレイの

時間と連結した機能のダイナミックなモニタリングの制御に関する。モニタされるデータは、ある種のパラメータがどのように設定されるかに基づいて変化する。図8で示されるように、ユーザに対して対話ボックスが、ディスプレイ画面上で提示されている。ユーザは、この画面を図7で示されるウィンドウのメニューまたはアクション・バー110から選択できる。この対話ボックス120は、ユーザがパラメータの新しい値を登録できるフィールド122を有する。ユーザが、新しい又は修正パラメータを打ち込むと、プログラムは新しいパラメータの値を使用するために根本的な機能を動的に修正する。これは、データ収集を制御するプログラムに対して、対話ボックスを制御するプログラムによるAPIコールを通して達成される。図9で示されるように、ユーザ・パラメータは、図8の画面の対話ボックスを通して、112で問いただされる。パラメータが有効であるかどうか、114で判定が行われる。もし有効でないならば、エラーメッセージは、116で表示され、ユーザのパラメータは、112で再び問いただされる。もし有効であるならば、新しいパラメータが、名前付きパイプ(図1の29)を経て118のデータ収集機構のAPIへ送り出される。データ収集機構は、データを受け、119で指定された機能のパラメータを変更する。

【0041】好ましい実施例では対話ボックスを使用しているが、他のタイプの制御が、ユーザから新しいパラメータを得るために同様に使用される。これらには、スクロール・バー、スピン・ボタン、エントリ・フィールドまたはコマンド・ライン・パラメータ等がある。

【0042】RAMモニタ・ウィンドウのRAMワーキング・セット・メモリのダイナミックなディスプレイに影響を及ぼすパラメータを変えるこの方法は、前述のRAMワーキング・セット期間を修正するのに使用される。これまで説明したように、RAMワーキング・セット・メモリは、ユーザが低いRAMワーキング・セット期間を選択すると低くなり、高いRAMワーキング・セット期間を選択すると高くなる。

【0043】周辺装置デバイス利用率について説明する。デバイスの利用率を求めるのに使われる一般の技術は、高レベルの解明タイミングを必要とせず、又は、ハードウェアおよび/又はオペレーティング・システムを変える必要はない。むしろ、この方法は、定期的にデバイスの状態をサンプリングし、そしてそのデバイスが“デバイスのビジー”状態を返す回数を記録する実行と実行の間を変えられるサンプリングの周期率を作り出す技術は、特定のものではなく、又は、根本的デバイス利用率の測定技術を理解するのに難しいものではない。例えば、パーソナル・コンピュータではハードウェア・タイマー割込みを使用すると好都合である。OS/2で実行するIBMパーソナル・コンピュータは、32ミリ秒毎に割込みが発生し、DOSを実行させているときは、5

5ミリ秒毎に発生する。さらに、デバイス状態を照会するのに利用される技術は、デバイスとデバイスとの間で変わるが、しかし、本発明の趣旨及び範囲内で他のタイプのデバイスに拡張できる。たとえば、IBMパーソナル・コンピュータESDIディスク駆動装置は、入力ポート・アドレス X'3512' (16進)で連続状態を提供する。他のデバイスでは、デバイスが状態情報を戻す前に、デバイス照会コマンドがそのデバイスに送り出されるのを必要とする。

10 【0044】ここで図10を参照するに、好ましい実施例のデバイス・ドライバである収集プログラム140は、ハードウェア・タイマー144から割り込み142を受ける。各タイマーのために、ボール・カウントが146で増分させられる。次に、測定されるデバイス148が、ビジーかまたは否定であるか求められるためにデバイス148は、連結されたデバイス・コントローラ150によって152で問いただされる。このビジー情報は、デバイス・コントローラ150によって154で報告される。報告されたデバイスがビジー状態かどうか、156でチェックされる。もしそうならば、ビジー・カウントが、158で増分させられる。収集は、tic142によって再びトリガされるまで終了する。

【0045】1度収集プログラムが、ユーザ指定のまたはデフォルト・パラメータによって決められた十分な数のサンプルを有すると、次に、報告プログラム162は、164で、そのビジーと全カウントを集め、そして166で、ビジー・カウントを全カウントで割ってデバイス利用率を計算する。この計算を下記に示す。

30 【0046】デバイス利用率 = ビジー・カウント / 全カウント

【0047】この利用率は、次に、以後で述べられるような、数字またはグラフィカル形式で168で、ディスプレイに書かれるような、いずれの方法によって報告でき、あるいはログ・ファイルに書込まれる。この報告プログラムは、好ましい実施例では定期的に収集プログラムのデバイス・ドライバを呼び出し、ビジーtic数のtic142のトータル数に対する比としてプロットする。デバイス・ドライバは、デバイス利用率を再計算するために1/2毎に呼び出されるが、しかし、この呼出しの頻度は、この好ましい実施例の説明で述べる手順でユーザが定義し、修正できる。

40 【0048】デバイス利用率は、直接測定されるよりむしろサンプリングによって評価されるので、その評価には潜在的なエラーが存在する。統計方法が、この潜在的エラーを予測できる。この分野の専門家は容易に理解されたように、ここで用いるサンプリング技術は、2つだけの可能な値、ビジーか、又はビジーでないかの、繰返しのサンプルを用いる。このようなサンプルは、ベルヌーイのサンプルと呼ばれ、2項分布に従う。さらに、サンプル数が比較的に大きい場合、例えば、20より大き

15

い場合は、2項分布は正規分布に近似する。正規分布においては、実際の比率と比較したサンプル比率のエラーは、ほとんど変わらない。

【0049】エラー =  $Z(a/2) * (x/n * (1 - x/n) / n) ** 1/2$

ここにおいて、

【0050】a = 所望の信頼水準（一般に 0.95 又は 0.99）

Z = 正規分布の標準ランダム変数

x = サンプルの成功数（この場合、ビジー・サンプル） 10

n = 全サンプル数

【0051】Z(a/2)の値は、統計表で見つけられる。例えば、信頼水準95%では、Z(a/2)は、1.960に等しい。例えば、信頼水準99%では、Z(a/2)は、2.576に等しい。

【0052】特定の例として、OS/2で実行しているIBMパーソナル・コンピュータを例にすると、 $10 * 1 / 0.032 = 312$ のサンプルの合計が10秒間で収集できる。さらに、最も大きい値を考慮してみる。x 20 が厳密にnの1/2である場合、 $(x/n * (1 - x/n))$ は、0.25を得る（これは初步の演算によって証明できる）信頼水準95%、10秒間で見つけられる最大のエラーにおいて、デバイス利用率の評価は越えない。

【0053】 $1.96 * (0.25 / 312) ** 1/2 = 0.055 = 5.5\%$

【0054】同様な計算では1分間のサンプルの最大エラーは、2.3%を示す。このように、統計値は、前述のデバイス・ビジー・サンプリング方法が、デバイス利用率の評価により正確度をもたらすことを示す。さらに、この方法は、従来のデバイス利用率情報を得る方法よりも簡単で低コストである。

【0055】周辺装置デバイス利用率を測定する代替方式を次に述べる。論理ディスク・アクティビティを測定するために、プロセスがAPIを通してファイル・システムをアクセスする際に生成される、ファイル・システム事象が、デバイス・ドライバ・セクションで述べた方法によってトレースされ整理される。

【0056】CPUアクティビティについて説明する。40 CPUアクティビティ、すなわち、利用率は、プロセスを開始させ、プロセスをシステムの最低の優先レベルに割り当てる好ましい実施例で測定される。本発明は、従来技術のプロセスが実行できる仕事量を追跡するのではなく、最低優先権のプロセスがシステムで実行する時間量を追跡する。最低レベルのプロセスは、高優先権の他のすべてのプロセスがそのタスクを完了し、CPUをもはや必要としない時に実行するだけなので、システムのアイドル状態の時間の量（または他のタスクを実行可能状態）は、アイドル・プロセスが実行した時間の量であ 50

16

る。好ましい実施例では、データ処理システムのタスクは、4つのクラスに分けられる。(I) タイム・クリティカル：最も高い優先権。(II) 固定high：何れの定型タスクの前に実行する。(III) レギュラ：アプリケーション・プログラムに割り当てられた通常のクラス。(IV) アイドル：タイム・クリティカル、固定high、又はレギュラの優先権タスクが実行準備で、実行しない段階である。

【0057】好ましい実施例では、OS/2オペレーティング・システムによって提供されるOS/2 RASトレース機構、すなわち、SYSTRACEは、低レベルのプロセス・システム・アクティビティの事象トレースを得るために使用される。このSYSTRACE機構に関しては、デバイス・ドライバ・セクションで詳細に述べられる。他のオペレーティング・システムによって提供される同様な他のタイプのシステム追跡機能は、本発明の趣旨及び範囲内で、このユーティリティがもたらす同様な方法で使用できる。以下は、特定のSYSTRACEユーティライゼーションの特徴を述べる。

【0058】デバイス・ドライバについて説明する。好ましい実施例では、デバイス・ドライバは、以下のSYSTRACEユーティリティを実行するためのものである。デバイス・ドライバは、通常の方法で取り付けられ、データ処理システムが読出す初期プログラム・ロード(IPL:Initial Program Load)のCONFIG.SYSファイルで識別される。フックと呼ばれる特別な命令グループは、実行フローを追跡するために、システムおよびアプリケーション・プログラムにおいて重要である。各フックは、独特な識別（主コードおよび従コード）を有し、他のフックとは区別され、データ項目としてキー・プログラム変数、記号またはリターン・コードを含む又は含まないこともある。OS/2の好ましい実施例では、フックが、生成され、収集され、そしてバッファ内に格納されるための手段を提供するSYSTRACEとして知られる機構が存在する。他のオペレーティング・システムは、自身のシステム・ユーティリティを使用して同様な機能性を提供する。このユーティリティは、フックを管理するための一般的なツールと見なされる。

【0059】デバイス・ドライバは、SYSTRACEを通るすべてのフックを捉え、そこに含まれる望まないフック又は情報をフィルタ・アウトし、制御プログラムによって所望された正味のフック及び情報だけを通過させる。デバイス・ドライバと制御プログラムは、前述のデータ収集機構を含む2つの素子である。

【0060】デバイス・ドライバ取り付け後、システム初期状態設定中に64Kのバッファが割り当てられ、データがフォーマットされ、制御プログラムに渡される。このバッファは、2つの32Kバッファに内部分割され、第2バッファは、デバイス・ドライバと制御プログ

17

ラム間の交信エリアとして使用される。交信エリアは、変数のために予約された単純なデータ処理システム・メモリの一部であるこのメモリは、デバイス・ドライバおよびアプリケーション・プログラムによってアクセス可\*

テーブル2

time_int equ	OFFEOH OFFEO & OFFE2を使用する第1DD変数
varA0 equ	OFFEOH タイムタグ算術に使用するワード
varA2 equ	OFFE2H タイムタグ算術に使用するワード
start_time equ	OFFE4H OFFE4 & OFFE6を使用する第2DD変数
varB0 equ	OFFE4H タイムタグ算術に使用するワード
varB2 equ	OFFE6H タイムタグ算術に使用するワード
elapsed_time equ	OFFE8H OFFE8 & OFFEAを使用する第3DD変数
var_FFE8 equ	OFFE8H タイムタグ算術に使用するワード
var_FFEA equ	OFFEAH タイムタグ算術に使用するワード
Dekko_SEL equ	OFFECH 第4DD: 1ワードだけのDEKKO第1OFFEC
PID equ	第4DD: PIDの他のワード
var_FFEC equ	OFFECH タイムタグ算術に使用するワード
var_FFEE equ	OFFEEH タイムタグ算術に使用するワード
flush equ	OFFOH 1ならば、フックをフラッシュする、そうでなければ通常に処理する
var_FFF0 equ	OFFOH タイムタグ算術に使用するワード
switch equ	OFFF2H 0でないならば、バッファをフラッシュ・バッファに切り替える
var_FFF2 equ	OFFF2H タイムタグ算術に使用するワード
reals equ	OFFF4H リアル・モードのフックの数を累算する
var_FFF4 equ	OFFF4H タイムタグ算術に使用するワード
var_FFF6 equ	OFFF6H タイムタグ算術に使用するワード
var_FFF8 equ	OFFF8H タイムタグ算術に使用するワード
int_nesting equ	OFFAH 割り込みの深さのネスティングを維持する
var_FFPA equ	OFFAH タイムタグ算術に使用するワード
current_time equ	OFFCH OFFFC & OFFFEを使用する最後のDD変数
oldtime equ	OFFCH 前の値をセーブする
bigtime equ	OFFEH 時間の高位ワードを持続
shortbuf equ	02020H 約24バイト有効なバッファのサイズは、08000 - shortbuf

18

\*能である。以下のテーブル2は、第2バッファ32Kの上位32ワード（すなわち最終部）を占める変数を定義する。

【0061】

【0062】フッキングについて説明する。このデバイス・ドライバのインストール中に、デバイス・ドライバは、将来の使用のためにラベル"strp\_common"に位置するオリジナルのSYSTRACEコードのコピーを

テーブル3

セーブする。OS/2システム・ルーチンDevHel pへのコールが、下に示されるように、このアドレスを得るために使用される。

【0063】

"strp\_common"の位置を得るためのサンプル・アセンブリ・コード

```

AX:BX 変数へのポイント
mov     al, 10D
mov     dl, DevHlp_GetDOSVar
call    DevHlp

```

【0064】暫くしてから、制御プログラムは、デバイス・ドライバが、SYSTRACE核コードの部分のパッチ（修正コード）をインストールする時に、デバイス・ドライバに“読出し”を実行する。パッチは、双峰（インテル・マイクロプロセッサ・アーキテクチャの一部で、当業者には周知の2つの異なるアドレス指定モードのリアル又はプロテクト・モード）コードを有する。双峰コードは、リアル又はプロテクトのいずれかのモードで、SYSTRACEを通して来るフックを捉えることができ、関心あるこれらのタスクをフィルタ・アウトして、事象の追跡のような他のタスクを実行する。

【0065】アンフッキングについて説明する。その後、システムが実行停止の準備に入る時、制御プログラ

#### テーブル4

##### 事象追跡例

```

Time_0Event_0 データ
Time_1Event_1 データ
Time_2Event_2 データ
Time_3Event_3 データ
Time_4Event_4 データ
Time_5Event_5 データ
.
.
.
Time_n-1Event_n-1 データ
Time_nEvent_n データ

```

【0068】SYSTRACE機構は、そのプロセスにおいて、タイムスタンプを事象に記録するために低分解能のシステム・クロックを使用する。これは、システム資源の性能の分析を試みるのに、本発明にとって不適當である。従って、データ処理システムのタイマーの1つが、事象間のデルタ（相違）時間を決めるのに使用され、及び前述のテーブル3のSYSTRACEレコードの時間を高分解能のタイムラグと取り替える。

【0069】タイマーについて説明する。好ましい実施例のハードウェア・タイマーは、インテル8253タイマーで、複数のタイマーを内蔵している。8253タイマーのさらに詳細な情報は、インテルのマニュアルの題名“インテル構成部品データ・カタログ”（Intel Component Data Catalogue）で述べられており、インテル社（Intel Literature Dept. in Santa Clara, CA.）から入手でき、ここでは、バックグラウンド材料として用いて

\*ムは、前にセーブされたSYSTRACE核コードがSYSTRACE機構の元の位置に再格納される時に、デバイス・ドライバに“書込み”を実行する。このように完全にオリジナルのSYSTRACE機能を復帰させる。

【0066】データの収集について説明する。事象の追跡は、データ処理システムで生じる事象を追跡するプロセスに関する。タイム・スタンプは、各事象と関係がある。事象は、格納され、年代順に処理される。事象が年代順なので、事象は、データ処理システムで発生する連続するアクティビティをもたらす。事象追跡の例が、以下のテーブル4で示される。

【0067】

いる。タイマー0は、モード2にプログラムされる。このモードは、高分解能のタイマーを提供し、0xFFFFで始まり、そして0x0000の下方へカウントして繰り返す。タイマー機能は、再び0xFFFFで始まる、すなわち、言い換えると、タイマーは回ることになる。タイマー3は、8253タイマー・モジュールの他のタイマーによって生成される通常の割り込みが禁止されるように、部分的に初期化される。通常、タイマー／カウンタの1つが、0をカウントしたとき、カウントされた時間が経過したことをシステムが知るように、ある割り込みが発生する。本発明の好ましい実施例では、タイマー／カウンタが所定の設定値に達した時に割り込みが生ずるのは不都合である。従って、部分的にタイマー3を初期化することによって割り込みの発生を禁止している。これはまた、ウォッチドッグ・タイマー（他の可能な実施例では、上記タイマーは、実現しているか、又

は、単にソフトウェア技術によってエミュレートされているだけである)として知られている。数えられる実際のタイム・インタバルは、1刻み、約0.8380953445マイクロ秒である。ここで図11を参照するに、システム・メイン・メモリ182に割り当てられ、0X0000に初期化される内部レジスタ180は、インタバル・タイマーが0X0000から0XFFFFに回る度に増分される。デバイス・ドライバは、タイマー・モジュール174の内蔵タイマー172から値を170で読出す。この値は、次に、その値が0X0000から0XFFFFの効果的な範囲になるように自己補正する。この補正されたタイマー値176は、内部レジスタ値178と組み合わせられ、1回りする前に約1時間ほどカウントできる32ビット・タイマー180を作る。16ビットの高位ワード178は、内部レジスタのロールオーバー・カウンタであり、及び低位ワード176も又16ビットで、補正タイマー値である。この32ビット値は、タイムタグ値180であり、下記で説明される。

【0070】タイミング保全性を維持するために、前述の内蔵タイマーは、ロールオーバーを間違わないように、少なくとも55ミリ秒毎に1度読出さなければならない。作動SYSTRACE主コード04は、この必要条件に十分である。主コード04は、タイマー割り込みを含めて、割り込みを許可又は可能にする好ましい実施例では、各タイマーの割り込みは、32ミリ秒毎に生ずるので、このことは、55ミリ秒毎に少なくとも1度の事象発生があっても(及び8253タイマーの連結された読出し)ミスのないことを保証することになる。これが8253タイマーが、割り込みを含めて、事象発生毎に読出される理由である。これで、タイマー・オペレーションが理解されたので、いつタイマーが読み出されるかに説明を向ける。

【0071】フックは、モニタ可能であり、及びデータ処理システムの特定の応答のトリガとなる事象である。OS/2での事象は、通常2つのフック、プレ・フック及びポスト・フックによって記述される。例えば、I/O要求が作られるとき、デバイス・ドライバは、要求がI/Oアダプタに作られようとしていることをシステムに合図するプレ・フックを生成する。アダプタがI/O要求を完了する際に、デバイス・ドライバは、ポスト・フックにその事象の終了を合図する。プレ・フックとポスト・フックとの間の時間は、その事象の経過時間を表す。特に、I/O要求のような事象が発生した場合、その事象を扱うカーネルは、その事象を述べる情報とともにSYSTRACEルーチンを呼び出す。これによって、SYSTRACEがその事象を処理することができる。フックがSYSTRACEバッチ・コードに到着する度に、すなわち、フックが呼び出され、SYSTRACEがそれを処理する度に、タイマーが読出され、高位バイトが必要に応じて増分される(すなわち、前述のよ

うに、タイマーが回る場合)。そのフックは、所望のフックの1つであるかどうか参照するために調べられる。受け取ったフックがモニタされるフックであるならば、さらに処理される。そうでなければ、フラッシュされ、あるいは通常の処理を続ける。

【0072】そのフックが割り込み(04/x x)であるならば、デバイス・ドライバが、割り込みの処理に費やされる時間を測定する。これは、プレ・フックを呼び出し、割り込みハンドラがその割り込み要求を処理し始めるとき生成される“割り込み開始”と呼ばれる事象、及びポスト・フックを呼び出し、割り込みハンドラが割り込み要求の処理を完了するとき生成される事象の“割り込み終了”とともに、マッチングすることによって実行される。このように、プレ・フックとポスト・フックには1:1の対応があり、及び各々のタイムスタンプは、割り込みを処理する時間を生むためにお互いから差し引かれる。

【0073】また、プレ・フックが生じた後、これに対応するポスト・フックが生じるまえに引き続き、プレ・フックが発生することも可能である。このフックのネスティングは、受けたいずれのポスト・フックが、最後に受けたプレ・フックとペアとなるようにすることで容易に取り扱われる。言い換えると、第1のフックの開始後、他のフックが開始できるが、しかし、第2のフックは、第1が終わる前に終了する。このネスティングのシナリオにおいて、終了時間から開始時間を差し引き、そして全ネスティング・アクティビティの所要時間を差し引くと、どれくらい外部の事象が時間を所要したかが分かる。

【0074】フックがモード・スイッチ(02/x x)であるならば、デバイス・ドライバは、第1モード切り替えからスケジューラが異なるプロセスをディスパッチするまでの時間を追跡することによってCPUのリアル・モードに費やした時間を測定する。この時間は、それからモードがプロテクト・モードに切り換わるまでの時間から差し引かれる。

【0075】フックがスレッド・ディスパッチ(12/01)であるならば、デバイス・ドライバは、最初に、そのデータ・エリアからプロセス識別(PID: process identification)及びスレッド識別(TID: thread identification)をセーブする(PID及びTIDは、OS/2構造システムで共通の用語である。PIDは、OS/2環境内のプロセスを独特に識別する16ビット番号である。PID値は、0001で始まり、プロセスが生成される毎に増分される。TIDは、複数のスレッドが1つのプロセスに存在するのに必須である)。次に、デバイス・ドライバは、前のスレッド・ディスパッチ・フック(12/01)と同じPIDがあるかどうか、データを調べ、肯定の場合はフックがフラッシュされる。否定の場合は、割り込みとリアル・モードで消費された時間

が、事象を述べるスケジューラによって提供されたPID及びTIDの既存のスレッド・ディスパッチ(12/01)データに加えられる。全スレッド・ディスパッチ・フックは、図12で示され後で説明される規格PERFMON/DEKOVERTフォーマットに従うために再フォーマットされ、デバイス・ドライバの32Kバッファの1つに書込まれる。累算された割り込み時間及び実モード時間を有する2つレジスタが、次にゼロにリセットされる。

【0076】フックがFile Systemフック(30/xx)であるならば、現在のTID191は、図13で示されるように通常データの前に挿入される。

【0077】上記掲載の、及び他の重要なフックは、また、PERFMON/DEKOVERTフォーマットとなるように再フォーマットされ、そして32Kバッファの1つに書込みされる。各レコードの最初の8バイトは、図12で示されるように、主コード183、副コード184、データ長185、フラグ186及び4バイトのタイムタグ188である。189で示される後続するバイトDD1~DDnは、フック・データである。バッファの1つがいっぱいになると、すなわち、32Kスペースの内24Kが使用されると、そのバッファは、従来のプログラミング技術を使用して、切り換えられ、全32Kバッファが、コントロール・プログラムに利用可能となる。データ収集は、他の32Kバッファに続く。

【0078】デバイス・ドライバは、また、それまで累算されたデータとともにコントロール・プログラムを提供できるように、コントロール・プログラムからの信号でこれらのバッファを交換できる。同様なオペレーションは、トレース・コマンド(00/02)が出された場合に生ずる。このとき、フックが受け取られて、第1データ・バイトは、0X00(トレースのオフを意味する)である。このケースにおいては、必然的にコントロール・プログラムが、直ちに現在のバッファを受け、いずれのバッファでも、これ以降、データの累算は行われない。

【0079】コントロール・プログラムとデバイス・ドライバ間の交信は、次に述べる通り、それぞれの32Kバッファで交信エリアを使用して実行される。コントロール・プログラムが、デバイス・ドライバ及びデバイス・ドライバのバッファをリセットできるように、コントロール・プログラムは、テーブル2で示されるように値'2'の交信エリアの制御ワードをロードする。デバイス・ドライバが、リセットを完了した時は、この値を'1'に変更する。制御プログラムが、デバイス・ドライバ及びバッファの休止を望む場合は、制御プログラムは、値'1'の通信エリアの制御ワードをロードする。制御プログラムが、デバイス・ドライバの再開を要求する場合は、制御プログラムは値'0'の制御ワードをロードする。コントロール・プログラムが休止又は停止を望む場

合は、コントロール・プログラムは、前述したようにデバイス・ドライバをアンフックする。オペレーションは、それから制御プログラムがデバイス・ドライバに他の読出しコマンドを送り出すまで中断される。

【0080】データ整理について説明する。データ整理低水準事象トレース性能データは、高水準システム・アクティビティに編成される。これは、以下の方法論によって達成される。最初に、プレ・フック及びポスト・フック事象が前述されたようにマッチングさせられ、次に、これらの2つのフックは、1つの事象に編成される。これは、その事象が、どれくらいの所要時間か分かっており、単一のレコードがプレ・フック及びポスト・フックを置き換えるために使用されるからである。そしてこの事象タイミングは、必要とされた情報の所望の細分性を有する。その上に、前述したように、事象は、コントロール・プログラムに重要な事象レコードにおいて、使用できる情報だけがフィルタリングされている。

【0081】アプリケーション・プログラミング・インタフェース(API:Application programming interface)について説明する。データ収集機構へのAPIについて述べる。このAPIによって、クライアント・アプリケーションが、性能データの検索及びメモリ・アナライザをアクセスすることができる。

【0082】APIは、システム・パイプ及びトレース・パイプと呼ばれる2つの名前付きパイプを通して実行される。システム・パイプは、データ収集機構に対してのパラメータの送受信に、クライアント・アプリケーションによって使用される。トレース・パイプは、連続性能データを受けるために、クライアント・アプリケーションによって使用される。データ収集機構は、両方のパイプをつくる。クライアント・アプリケーションは、パイプにアクセスするためにOS/2機能コールのDos Open及びDos Closeを呼び出す。両パイプは、ブロッキング・モードでメッセージ・パイプとして作られる("IBMオペレーティング・システム/2、バージョン1.2、プログラミング・ツール及び情報"(IBM Operating System/2 Version 1.2 Programming Tools and Information)を詳細な情報として参照されたい。ここではバックグラウンド材料として参照している)。

【0083】システム・パイプについて説明する。クライアント・アプリケーションは、システム・パイプを通してデータ収集機構のアクションを制御する。クライアント・アプリケーションは、パイプにメッセージ・モードで読書きする。パイプに書込まれた各メッセージは、構文図表からの1つのパラメータを表す。メッセージは、数字(例えば、10進の10は、ストリング"10"のように送られなければならない)を含むASCII Zストリング(つまり、空白での終了、すなわち、2進のゼロの1バイト)でなければならない。

【0084】データ収集機構は、システム・パイプを通してクライアント・アプリケーションへ応答を送り返す。ローカル・マシンのシステム・パイプのOS/2機能コールDosOpenのクライアント・アプリケーションによって使用される名前は、\PIPE\SYSTEM.SPMである。リモート・サーバでのパイプの名は、\\server\_name\PIPE\SYSTEM.SPMである。

【0085】メモリ・アナライザ (/THESEUS theseus\_command) からの出力は、また、システム・パイプを通してデータ収集機構からクライアント・アプリケーションに送り出される。最初に、リターン・コードが送り出される。それから、メモリ・アナライザからの出力があれば、コマンドが送り出される。メモリ・アナライザからの各メッセージは、1行で表される。最大ライン長さは、100文字である。この出力の後には、空白キャラクタ(00)が続く5つの符号(#####)によって

テーブル5 データ収集機構パラメータ

パラメータ アクション

/START トレース・パイプ・レコードが、データ収集機構によって送り出される資源の型を示す。また、テーブル8-3を参照する。

\* /STARTパラメータと共に使用されるときCPU、物理ディスク、RAM及びスワップ資源を示す(論理ディスクは示さない)。/STOPパラメータと共に使用されるとき、すべての資源を示す。

CPU CPU資源を示す。

PHYSICALDISK

物理ディスクの資源を示す。

LOGICALDISK

論理ディスク資源を示す。

RAM ランダム・アクセス・メモリ資源を示す。

SWAP スワッピング資源を示す

注意: いずれかの前のオプションが、指定される場合、指定タイプ(テーブル8-3で"No Type"として指定された)以外の全トレース・パイプ・レコードが含まれる。

パラメータ アクション

/STOP トレース・パイプ・レコードが、データ収集機構によって送り出されない資源のタイプを示す。/STARTのオプション記述を参照する。また、テーブル8-3も参照する。

##### 実行メッセージ。/START又は/STOPパラメータが続く、資源明細メッセージの終了を示す。

/COMMENT

収集データに注釈を埋め込む。

string 現在の収集データに埋め込まれる注釈。コメントは、40文字以内。長いコメントは、40文字に先頭部を除去され、データ収集機構によってエラーなしで受けられる。ストリングが、埋め込ま

表される実行メッセージが続く。システム・パイプは、クライアント・アプリケーションが、OS/2機能コールDosCloseとともにパイプを閉じるときデータ収集機構によって切り離される。

【0086】図14は、システム・パイプを通してデータ収集機構に送り出されるメッセージの構文図表を説明する。構文パラメータは、テーブル5で説明される。パラメータは、文字の連続セットとして表される。大文字のキャラクタのパラメータは、キーワードである。

【0087】各パラメータは、空白文字(2進ゼロ)で終了しなければならない。例えば、注釈"Performance data for SERVER1"をSPMアプリケーションに送るには、以下のメッセージを送る:

【0088】/COMMENT

Performance data for SERVER 1

れたブランクを含むならば、ストリングは、独立したメッセージとして送り出される。

---

／EXIT データの受信を停止し、メモリからデータ収集機構を解放する。データ収集機構によって開始していた全プロセスもまた、停止する（IDLECPU及びTHESEUS）。

---

#### ／INITDATA

トレース・パイプを通してデータ収集機構から初期化レコードが送られる。以下のレコードが含まれる：

- － IDLECPUプロセスに対してのプロセス情報レコード。これは、CPUがアイドルの時の時間を求めるのに、データ収集機構によって使用されるプロセスである。このプロセスは、アイドル優先権レベル0（ゼロ）で実行する。
- － システム情報レコード。
- － システムで現在実行中の全処理のプロセス情報レコードこれらのレコードは、メモリ・アナライザが開始している時だけ、送り出される（／THESEUS STARTパラメータを参照）

注意：CPU資源は、プロセス情報レコードを得るために開始（テーブル8-3の／START CPUパラメータを参照）しなければならない。

---

パラメータ アクション

／TOD トレース・パイプを通してデータ収集機構によって送り出された日付レコード間のインタバル（秒）を指定する。

interval 日付レコード間の秒数可能な値の範囲は、1～100秒である。デフォルトは、5秒であるが、しかし、intervalパラメータは、従来のアプリケーションによって他の値に設定できる。

---

／RAM サンプリング・ランダム・アクセス・メモリで使用される期間を制御する。サンプルの情報に関しては、テーブル8-3のRAMレコード記述を参照する。

注意：このパラメータは、／START RAMを意味しない。RAM資源を可能にする詳細な情報は、テーブルの冒頭の／START RAMを参照する。

working\_set\_period

ワーキング・セット内の物理的RAMの量を求めるのに使用される秒単位のタイム・フレーム。各サンプルは、最後のワーキング・セット期間中に使用されたRAMの量を表す。可能な値の範囲は、5～3600である。デフォルトは、60秒であるが、しかし、working\_set\_periodパラメータは、従来のアプリケーションによって他の値に設定できる。

注意：ワーキング・セット期間全体が経過するまで、ワーキング・セットは、ワーキング・セットのRAMの比率だけを表す。これは、／RAMパラメータを出しているのか、又はワーキング・セット期間が変わったかの理由による。

sample\_interval

RAMサンプル間の秒数である。サンプルが実行される度に、RAMトレース・パイプ・レコードが送り出される。可能な値の範囲は、5～3600である。デフォルトは10秒であるが、しか

し、sample\_interval パラメータは、従来のアプリケーションによって他の値に設定できる。

注意：性能上の理由から、SPMアプリケーションは、working\_set\_periodパラメータ値をsample\_interval パラメータ値で割った値は、200以下を必要条件とする。

#### パラメータ アクション /THESEUS

メモリ・アナライザが、データ収集機構によって開始されていなければ、メモリ・アナライザを開始する。プログラミング・インタフェースをアプリケーションからメモリ・アナライザに提供する。

注意：メモリ・アナライザ全画面インタフェースは、この様に、データ収集機構によって開始するメモリ・アナライザのコピーからは利用できない。

START メモリ・アナライザが、データ収集機構によって開始させられていない場合、メモリ・アナライザを開始させる。

#### theseus\_command

いずれの有効なメモリ・アナライザ・コマンド。theseus\_commandは、コマンドがブランクを含むならば、独立したメッセージとして送り出されなければならない。

注意：全てのメモリ・アナライザ・コマンド (theseus\_command) は、THESEUS LOG コマンドを含めて、メモリ・アナライザによって直接解読される。全アクションが、メモリ・アナライザ全画面インタフェースでタイプされたかのように、データ収集機構によって開始させられたメモリ・アナライザの基準ポイントから起こる。

#### パラメータ アクション /NOTHESEUS

メモリ・アナライザが、データ収集機構によって開始させられた場合、メモリ・アナライザを終了する。これは、収集マシンのメモリ・アナライザと連結されたRAMオーバヘッドをセーブする。しかしながら、現在システムで実行している処理 (IDLECPU プロセスは別として) のプロセス情報レコードは、トレース・パイプを通して送り出されない。これは、/INITINFOパラメータが送り出されるとき実行しているプロセスを含む。

/DEBUG データ収集機構が、クライアント・アプリケーションからワーキング・ディレクトリのログ・ファイルSPMLOG. LOGに受け取るログ・パラメータに存在することを示す。

【0089】最初の文字にスラッシュ (/) を有する全てのパラメータのために1つのステータス・メッセージが、データ収集機構によってクライアント・アプリケーションに送り出されるこのステータス・メッセージは、パラメータからの要求の成功の徴候を提供する。ステータス・メッセージのフォーマットが、以下のテーブルで記述される。

【0090】

SPMリターン・コード	2バイト (ワード)
サービス・リターン・コード	2バイト (ワード)
確保した	2バイト (ワード)

【0091】次に、/THESEUSパラメータ (theseus\_command) で指定されたいずれのメモリ・アナライザ・コマンドからの出力が、クライアント・アプリケーションに送り出される。各メッセージは、メモリ・アナ

ライザからの1行を表す。実行メッセージ (### #

#)はこの出力に続く。

値は、16進で与えられる。

【0092】SPMリターン・コード・フィールドにおいて復帰できる値が、テーブル6に含まれる。すべての

【0093】

テーブル6 SPMリターン・コード

コード	記述
X'0000'	エラーなし、パラメータは受け入れられた。
X'0007'	不当なパラメータ。サービス・リターン・コードは、失敗したパラメータの一連番号を有する。スラッシュ (/) で始まる各パラメータは、その一連番号を1にリセットする。
X'0010'	working_set_period値が、範囲外である (/RAMパラメータ)。
X'0011'	sample_interval値が、範囲外である (/RAMパラメータ)。
X'0012'	sample_interval値は、複数のworking_set_period値でない (/RAMパラメータ)。
X'0013'	sample_interval値で割られたworking_set_period 値は、200より大きい (/RAMパラメータ)。
X'0014'	/TODインタバル値が、範囲外である。
X'0108'	OS/2機能コールDosExecPgmを通してOS/2システムにTRACE.EXEオン・コマンドを出すことができない。
X'0208'	DosExecPgm を通してOS/2システムにTRACE.EXEオフコマンドを出すことができない。
X'0408'	OS/2機能コールDosKillProcess を通してIDLESPU.EXEプログラムを始めることができない。
X'0409'	OS/2機能コールDosKillProcess を通してIDLESPU.EXEプログラムを止めることができない。
X'0806'	メモリ・アナライザが、このOS/2バージョンを認めない。
X'0807'	メモリ・アナライザと通信できない。
X'0808'	OS/2機能コールDosExecPgm を通してTHESEUS.EXEプログラムを始めることができない。
X'0809'	OS/2機能コールDosKillProcess を通してTHESEUS.EXEプログラムを止めることができない。
コード	記述

33

34

X'1003' デバイス駆動機構THESEUS.SYSは、CONFIG.SYS ファイルからロードされなかった。

X'1005' デバイス・ドライバTHESEUS.SYSの不当なバージョンが、CONFIG.SYSファイルからロードされなかった。

X'2003' デバイス・ドライバSPMDCF.SYSは、CONFIG.SYSファイルの中で行方不明である。

X'2004' エラーが、DosOpen又はDosReadを通してSPMDCF.SYSデバイス・ドライバを初期化中に発生した。

注意： サービス・リターン・コードは、このテーブルで別な方法で述べない限り、要求されたOS/2サービスからのリターン・コードである。

【0094】トレース・パイプについて説明する。トレース・パイプは、データ収集機構から性能データを検索するためにクライアント・アプリケーションによって使用される。トレース・パイプは、一方通行の名前付きパイプである。すなわち、クライアント・アプリケーションへのデータ収集機構である。単体のマシン上に、DosOpen機能コールのクライアント・アプリケーションによって使用されるパイプの名は、\PIPE\TRACE.SPNで、リモート・サーバでのパイプ名は、\\server\_name\PIPE\TRACE.SPNである。トレース・パイプは、最大メッセージ長さが8キロバイトのメッセージ・ストリーム名前付きパイプ（バイト・ストリーム名前付きパイプに対して）である。クライアント・アプリケーションは、トレース・パイプを通して性能データの収集及び伝送を停止させるためには、システム・パイプに対して/STOP又は/EXITメッセージを送らねばならない。

【0095】データは、トレース・パイプを通して伝達する前にデータ収集機構によってバッファ内で待ち行列に入れられる。トレース・パイプのメッセージは、1つ以上の完全なトレース・パイプ・レコードを有する。デ\*

\*ータが利用可能ならば、メッセージは、パイプを通して少くとも4秒毎に伝送される。

【0096】クライアント・アプリケーションが、データ収集機構から性能データを収集する連続アクションは次の通りである。

1. システム・パイプをオープンする。
2. システム・パイプを通して適切なメッセージをデータ収集機構（SPMDCF）に送り（/STARTメッセージを含む）、適用できるリターン・コードを得る。
3. トレース・パイプをオープンする。
4. データ収集の停止が用意されるまで、トレース・パイプからデータを読み出す。
5. システム・パイプを通してデータ収集機構へ/STOP又は/EXITメッセージを送信し、適用できるリターン・コードを得る。
6. システム及びトレース・パイプをクローズする。

【0097】トレース・パイプのレコード・フォーマットを説明する。SPMトレース・パイプを通して送られる一般のレコード・フォーマットは、

【0098】

レコード長 (1バイト)	トレース・パイプ・コード (1バイト)	調整可能なデータ長 (最大250バイト)
-----------------	------------------------	-------------------------

【0099】トレース・パイプ・レコードを説明する。図15～図17にリストされたレコードは、トレース・パイプを通してSPMアプリケーションからクライアント・アプリケーションに送られる。

【0100】テーブル定義  
ASCII Zストリング  
文字のストリングで、後に空白（ASCII 00）が続く。最大文字数は、250文字である。

【0101】データ・オーバーフロー

データがデータ収集機構によって破棄されたことを示す。通常、これは、クライアント・アプリケーションのトレース・パイプからのデータ読出しの速度が十分でない場合に生ずる。

【0102】ダブルワード

インテルのフォーマットで4バイト（すなわち、バイト／ワード確保）IBM C/2では、これは、無記名の長い整数（ULONG）である。

50 【0103】経過時間

オペレーション中に生ずる全タイマー的時間測定。これは、CPUを使用しているオペレーションが、ビジーであった時間と解釈しない。むしろ要求がいつ提出されたか、及びいつオペレーションが完了したかの間の時間である。例えば、スワップがスワッパーによって要求されると、次に、スワッパーは、ディスクI/Oが完了するまで、CPUを他のプロセスに引き渡す。それから、スワッパーは、オペレーションを完了する。経過時間には、スワッパーがディスクを待つ間の中断時間を含む、全時間が含まれる。

#### 【0104】第1物理ディスクのID

IDは、システムによって第1物理ディスクに割り当てられる。各物理ディスクは、第1物理ディスクに割り当てられたIDから始まって、連続番号を割り当てられる。

#### 【0105】物理ディスクの数

システムに取り付けられた物理ディスクの全体の数。

#### 【0106】セクタ数

512バイトのセクタの数。

#### 【0107】物理ディスクID

物理ディスクに割り当てられたID。

#### 【0108】プロセス名

これは、.EXEヘッダで定義されたプロセス名又は、EXEファイル（ピリオド又はファイル拡張を含まない）のファイル名である。

#### 【0109】従来のプロセスの実行タイム

従来のプロセスを実行している間に生じる全体的なタイマー（割り込みレベルで消費された時間を含む【従来のプロセスの割り込み所要時間】）。

#### 【0110】従来のプロセスの割り込み所要時間

従来のプロセスを実行中の割り込みレベルで生じる全体的なタイマー。

【0111】1日のレコードの最後のタイムからの時間最後のレコードが送り出された以降のタイマーの経過時間。この値は、正確な計算のために提供される。

#### 【0112】タイマティック

値は、8253/8254チップから得る。この値は、0.8380953445を掛けることによってマイクロ秒に変換される。すなわち、マイクロ秒 = タイマティック × 0.8380953445

#### 【0113】TRACECMD

ユーザがトレース・コマンドを出したことを示す。

#### 【0114】ワード

インテルのフォーマットでは2バイト（すなわち、バイト確保）。IBM C/2では、無記名の短い整数（USHORT）である。

【0115】グラフィカルな提示について説明する。資源利用率及び前述の性能モニタをグラフィカルに表示するために、本発明の好ましい実施例は、OS/2提示管理ウィンドウ及びグラフィックス機能を使用する。この

方法によって、ユーザは、関連する情報の複数のグループを複数のウィンドウ（又はビューポート）で同時に考察することができる。1つの主ウィンドウは、親ウィンドウと呼ばれ、子ウィンドウと呼ばれるすべての他のウィンドウを有する。これらのウィンドウが資源利用率の情報を表示する。図6で示されるように、資源情報が、ある種のデータ処理システム資源の利用率を表すグラフ形式で提示されている。資源利用率データは、ユーザが設定可能な時間（例えば、最大600秒間）を提示して、すなわち、図8の122で示されるサンプリング期間を提示して表示される。従って、瞬間、及び現在/過去の資源利用率のレコードを提供する。ユーザは、全ての又は一部の資源モニタの表示を選択でき、又、ウィンドウの表示文字を修正できる。他の情報、又は同一情報を他の形式で子ウィンドウで提示できる。どのように及びいつ、データの子ウィンドウに表示するかを制御する提示パラメータは、ユーザが主ウィンドウのメニュー（アクション）バーで変更できる。当業者が理解されているように、標準ウィンドウズ・プログラミング技術が、所望のグラフィカル表現を提示するために、OS/2提示管理インタフェースに使用されている。提示管理は、データを表示ウィンドウ又はビューポートに実際に提示するエンティティである。他のオペレーティング・システムには、例えば、Microsoft's Window2 (Microsoft社の登録商標)、HP's New Wave3 (Hewlett-Packard社の登録商標)、XWindows4 (M.I.T.の登録商標)、又は、AIXWindows5 (IBM社の登録商標)とDOSとの組み合わせ等がある。これらは、本発明の趣旨及び範囲内で、ウィンドウ同様な提示に同じようなプログラミング・インタフェースを備えることができ、そして同様にそれぞれのシステムで資源モニタの提示に使用できる。

【0116】最後に、図18は、本発明の好ましい実施例に使用した概略的なデータ処理システムを示す。CPU190、RAM194及び周辺装置デバイス196（直接アクセス記憶装置デバイス、すなわち、DASDとして示されている）のこれらは、バス構造を通して相互接続されている。同様に、ROS192及びポインタ/入力デバイス200を有するキーボード198が、このバス204に取り付けられている。これらは、好ましい実施例でモニタ可能な資源である。また、このバスにはユーザに資源モニタの結果を提示できるディスプレイ手段202が付けられている。このディスプレイ手段は、同様に共通バス204に付けられる。特定のデバイス間の高速バス及び図示されている通常のバス以外のバスを含む他の変更は、本発明の領域内であり、本発明のクレームの趣旨及び範囲内である。

#### 【0117】

【発明の効果】前述のように、トレース付きのこのデータ処理システム利用率モニタは、ハードウェアの特殊化なしで、及びモニタされるシステム性能に重大な影響を

37

与えることなしに、リアル・タイムの性能モニタリングを提供する。

【0118】本発明の好ましい実施例に関して説明を行ったが、これらは、ここで開示した正確な構造を制限するものではなく、及び本発明の範囲内での全ての変更及び修正の権利を保留する。

【0119】付録A-1～A-13は、APIにインタフェースするためのサンプルCソース・コードである。

#### 付録A-1

##### SPM API Cサンプル・プログラム

プログラムの目的: このプログラムは、SPMシステム・パイプをオープンし、SPM APIを使用してコマンドをSPMデータ収集機構に送る。特に、プログラムは、データ収集機構に対してスワッピング・アクティビティを報告するように命令する。次に、プログラムは、トレース・パイプをオープンしてレコードを読出す。このプログラムは、トレース・パイプからのスワップ・レコードを使用し、及びこれらのレコードを標準出力装置に表示する。ユーザが、F3キーを押すと、プログラムはSPMデータ収集機構にデータの送り出しの停止を命令し、及びプログラムは、パイプおよび出口をクローズする。プログラムのデモンストレーション。このプログラムは、アプリケーション・プログラミング・インタフェース (API: Application Programming Interface) を通してSPMデータ収集機構との通信例を提供する。プログラムは、SPMシステム・パイプおよびトレース・パイプ使用の1用例を示す。

ツールキット・ヘッダ・ファイルの必須セクションの記載

構造をバイト境界で実行させるIBM C/2の語用

#### 付録A-2

##### 必須ヘッダ・ファイルの記載

##### グローバル定義

##### SPMシステム・パイプ・リターン・コードの構造

##### プロトタイプ定義

##### SPMDCFと通信開始

##### SPMDCFとの通信停止

##### 性能データの読出し

##### 名前付きパイプをオープンする

##### 名前付きパイプをクローズする

##### 名前付きパイプからデータの読出し

##### 名前付きパイプへのデータの書き込み

##### 主手続きの開始

##### 主要部

メインが、機能呼び出す1) SPMデータ収集機構と通信開始を行い、2) SPMトレース・パイプからのデータを読出して翻訳し、3) SPMデータ収集機構との通信をクローズする。トレース・パイプ及びシステム・パイプは、ヘッダファイルで定義され、初期化される

#### 付録A-3

38

SPMデータ収集機構にデータを送るように命令する

データを読出し、所望の情報を出力する

SPMデータ収集機構にデータの送り出しの停止を命令する

主手続きの終了

InitSPMDCF: SPMデータ収集機構と通信開始する。システム・パイプをオープンし、スワッピング・アクティビティについての性能データを送るよう命令し、次に、トレース・パイプをオープンする。エラーが生じた場合、OS/2サービス・リターン・コードに復帰する。

SPMDCFリターン・コード構造

#### 付録A-4

システム・パイプをオープンしてSPMデータ収集機構と通信を開始する。スワッピングが検知された場合、SPMデータ収集機構にトレース・パイプのレコードを送るよう命令する。

SPMデータ収集機構にトレース・パイプの上のレコードを送るよう命令する

SPMデータ収集機構に対してこのメッセージが完了したことを告げる。SPMデータ収集機構からの応答を読み、チェックする

#### 付録A-5

SPMDCFからのデータを読むためにトレース・パイプをオープンする

##### ReadTraceData

この機能は、スワッピングが発生することを指示するレコードについてトレース・パイプをモニタする。そのようなレコードがトレース・パイプから読出される場合、レコードはプリントされる。F3キーを押すと機能が終了する

トレース・データのバッファ

トレース・バッファの位置ホルダ

バッファのレコードのポインタ

名前付きパイプ・ファイル・ハンドルのローカルの記憶装置

ローカルの記憶装置 (高性能)

F3キーが、押されるまでトレース・パイプからデータを読出す; データを受信するまで呼び出す。ERROR\_MORE\_DATAならば、新しいバッファを読出す。

#### 付録A-6

ERROR\_MORE\_DATAは、予期されない、異常環境で発生する。

エラーをチェックする

そうである場合、終了

現在のトレース・バッファを指示するように、トレース・バッファ・ポインタをセットする

バッファ内の現在のレコードを指示するように、トレース・レコード・ポインタをセットする

所望するトレース・レコードのためにバッファをスキャンする

送り `spmRecord` ポインタ

スプリアス・トレース・レコード長のチェック

このエラーは、通常の条件下では発生してはならない。

付録A-7

レコードの種類を翻訳し、コンパイルを開始する

このサンプル・プログラムでは、スワップ・レコードだけをシークする

レコードをスワップ・インする

レコードをスワップ・アウトする

このレコードを無視する

切り替えて終了

そのまま終了

付録A-8

ユーザが放棄 (F3キー) するかどうか確認する

キーストロークを待たない

F3キーが押された

`StopSPMDCF`: SPMデータ収集機構にトレース・パイプの性能データの送信を停止するように命令する。次に、トレース及びシステム・パイプをクローズする。

SPMデータ収集機構にレコードの送信停止を命令する  
SPMデータ収集機構にメッセージが完了したことを告げる。

付録A-9

SPMデータ収集機構からの応答を読み、チェックする

トレース・パイプをクローズする

システム・パイプをクローズする

付録A-10

`OpenPipe`: パラメータ・パイプによって指示されたパイプをオープンする試みを行う。パイプがビジーならば、この手順は、待機し、パイプがフリーになってから再びそのパイプのオープンを試みる。パイプが再びビジーならば、次に、`ERROR_PIPE_BUSY`のリターン・コードが復帰される。エラーが発生した場合、`DosOpen`リターン・コードが復帰される。すでにパイプがオープン状態であるならば、オープンを実行しない

そうである場合は終了

オープンしようとしている名前付きパイプに対して、正しいファイル・モードをセットする

システム・パイプ

トレース・パイプ

パイプのオープンを試みる

付録A-11

ファイル属性

オープン・フラグ = 生成又はオープンする

オープン・モード

確保された

2回試みて、エラー・リターン・コードを得たならば、エラー・リターン・コードを復帰する。

リターン・コードで実行する

パイプが、オープンしている、フラグをセットする

名前付きパイプが、所望する状態であるか、確認する

名前付きパイプが、フリー又は時間切れまで待機する

エラーが発生した場合、`DosOpen`からリターン・コードを戻す

付録A-12

10 切り替えて終了

1度の`DosOpen`が試みられたか、又はエラーが発生しない場合以外は、終了する

試みが失敗したことを示すフラグをセットする

終了する

`ClosePipe`: パラメータによって示されたパイプをクローズし、そして`DosClose`リターン・コードを復帰する。パイプ構造のパイプ状態を設定する。

そうであれば終了する

付録A-13

20 `WritePipe`: データをパラメータ・パイプによって示されたパイプに書き込みする。データは、バッファに格納され、及びデータ長は、`BufLen`に格納される。データが、`ASCIIZ`ストリングであるならば、`BufLen`をゼロにセットすると、`WritePipe`がそのストリング長を計算する。`DosWrite`リターン・コードは、データ・バッファよりも少ないデータで書かれた以外は、復帰され、`ERROR_MORE_DATA`が、復帰される。`DosWrite`エラーが発生したならば、書込まれたバイト数は、ゼロにリセットする。指定がない場合は、バッファの長さを計算する

`ASCIIZ`ストリング

そうでなければ終了する

データをパイプに書く

書き込みが、成功したかチェックする

データの一部分が、書込まれなかった

そうであれば、終了する

付録A-14

`ReadPipe`: パラメータによって指示されたパイプを読出す。データは、パイプから指定されたバッファに読込まれ、データの長さは、パイプ構造に戻される。`BufLen`は、データ・バッファのサイズを示す。`DosClose`リターン・コードを復帰する。エラーが発生したならば、読出されたバイト数をゼロに設定する。

`SAMPLE C` サンプル・プログラム・ヘッダ・ファイル (.H)

`SAMPLE`ヘッダ・ファイルは、`SAMPLE.C`ファイルで使用される記号の定数を定義する。`SAMPLE`ローカル手順宣言は、これらが使用される前に宣言さ

50 れたことを確実にするためにこのファイルに現れる。

## 付録A-15

DosCreateNmPipeに対してのオープン・モード

DosOpenに対してのオープン・モード

OpenMode 0010000011000000  
0010

DosMakeNmPipeに対してのパイプ・モード

読取り書き込み要求のデータ待つ

構造およびタイプの定義

PIPEDEF構造は、パイプに関する情報を所有する

## 付録A-16

SPMシステム・パイプ (SPMDCFとアプリケーションとの通信)

実行時間ステータス・フィールド

オープン・モード

データの送受信

パイプ・モード

出力バッファサイズ

入力バッファサイズ

パイプがビジーの場合、ミリ秒ほど待機する

パイプの名前

パイプ・バッファへのポインタ

SPMトレース・パイプ (SPMDCFからアプリケーションに送り出したトレース・レコード)

実行時間ステータス・フィールド

オープン・モード

データ専用読出し

パイプ・モード

出力バッファサイズ

入力バッファサイズ

パイプがビジーの場合、ミリ秒ほど待機する

パイプの名前

パイプ・バッファへのポインタ

【図面の簡単な説明】

【図1】システム性能モニタの概念モデルを示す図であ

る。

【図2】システム性能モニタの機能モデルを示す図である。

【図3】モニタされるシステム・メモリのカテゴリを示す図である。

【図4】ワーキング・セット計算タイミングを説明する図である。

【図5】システム・メモリ使用アルゴリズムの流れ図である。

【図6】測定されるシステム資源のグラフィカルな表示を示す図である。

【図7】ビューポート・メニューの使用を示す図である。

【図8】ビューポート・メニューの使用を示す図である。

【図9】ユーザ入力及び更新パラメータの受信の流れ図である。

【図10】周辺装置デバイス利用率がどのように測定されるかを示す図である。

20 【図11】高分解能システム・タイマーの構築を示す図である。

【図12】内部デバイス・ドライバ・バッファに格納される一般のレコードのフォーマットを示す図である。

【図13】内部デバイス・ドライバ・バッファに格納される一般のレコードのフォーマットを示す図である。

【図14】性能データ収集制御プログラムにインタフェースするアプリケーション・プログラミングのための構文構成を示す図である。

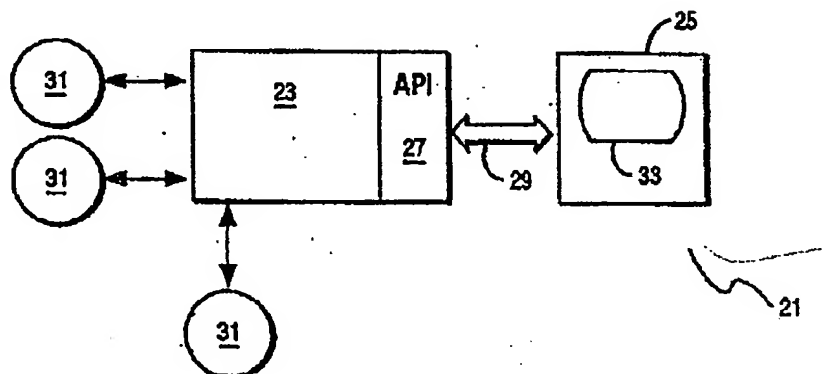
30 【図15】トレース・パイプ・レコードを説明する図である。

【図16】トレース・パイプ・レコードを説明する図である。

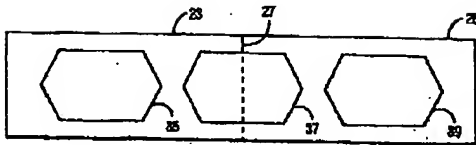
【図17】トレース・パイプ・レコードを説明する図である。

【図18】データ処理システムを示す図である。

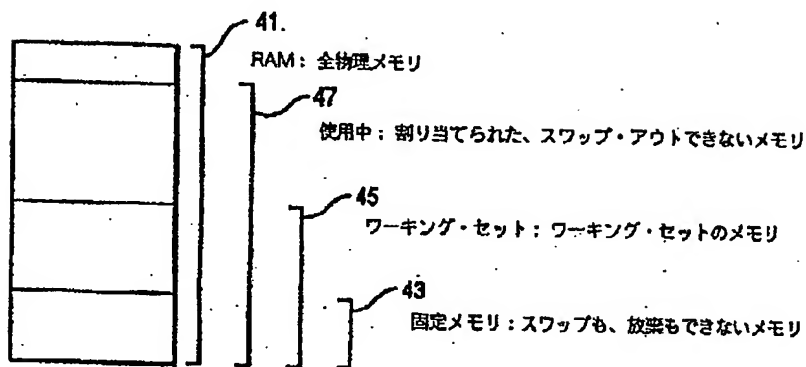
【図1】



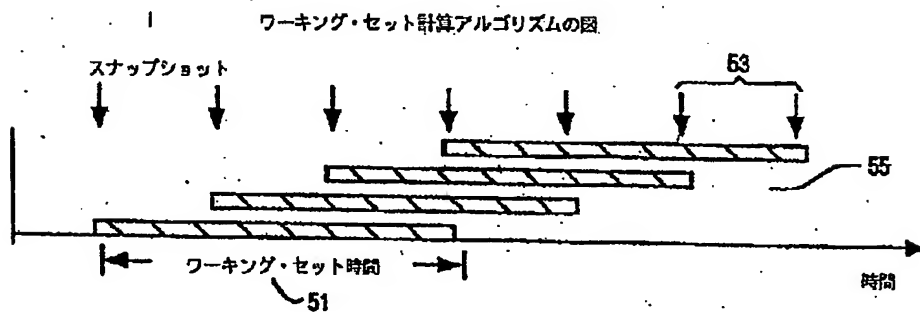
【図2】



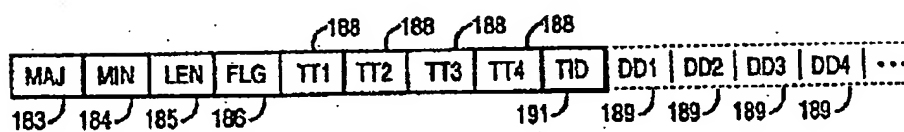
【図3】



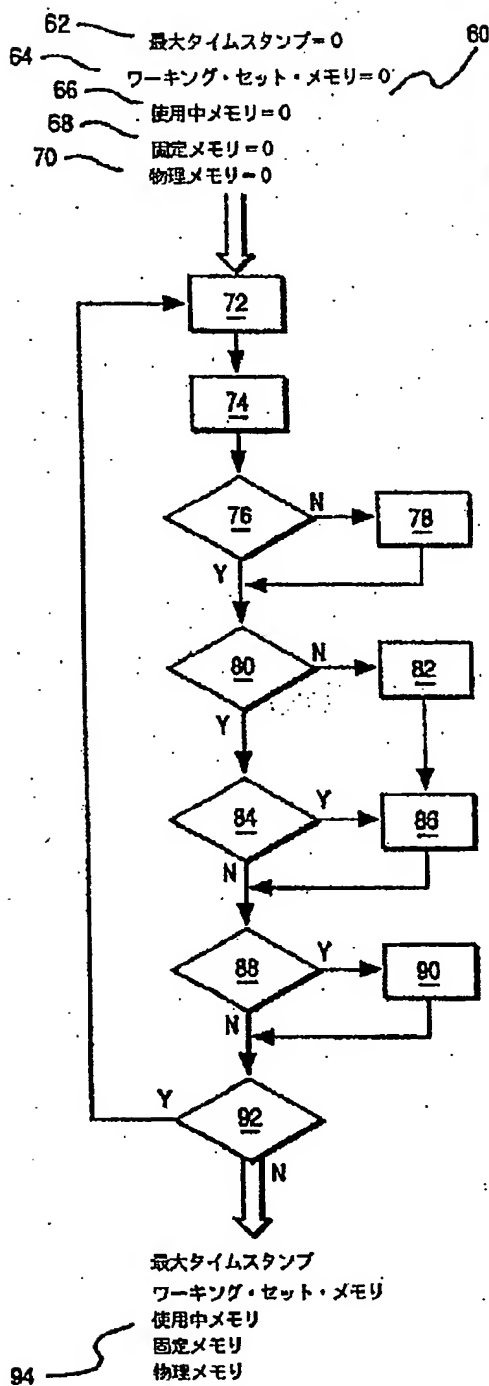
【図4】



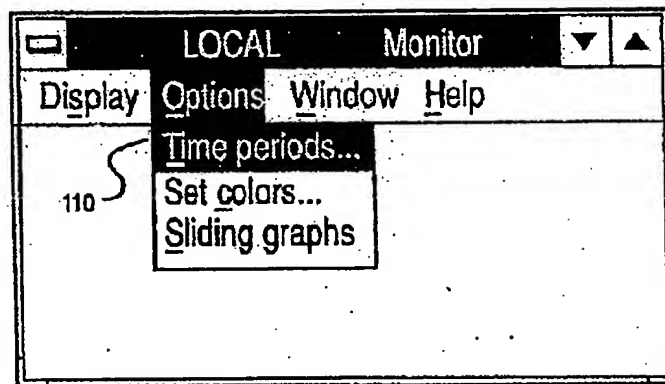
【図13】



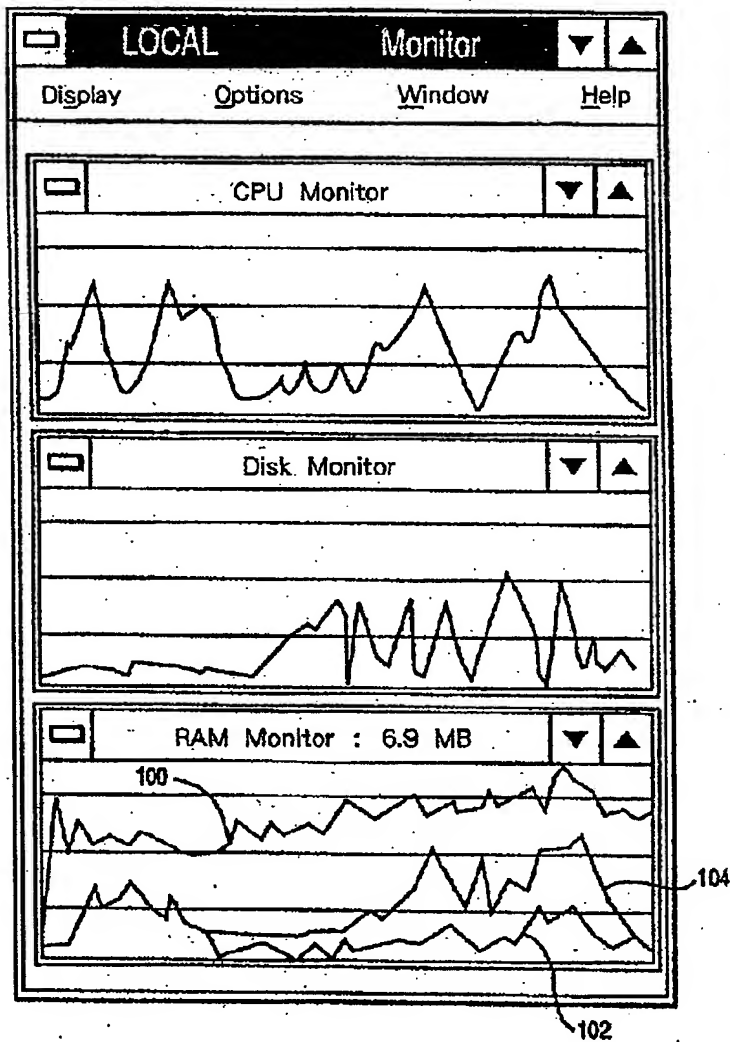
【図5】



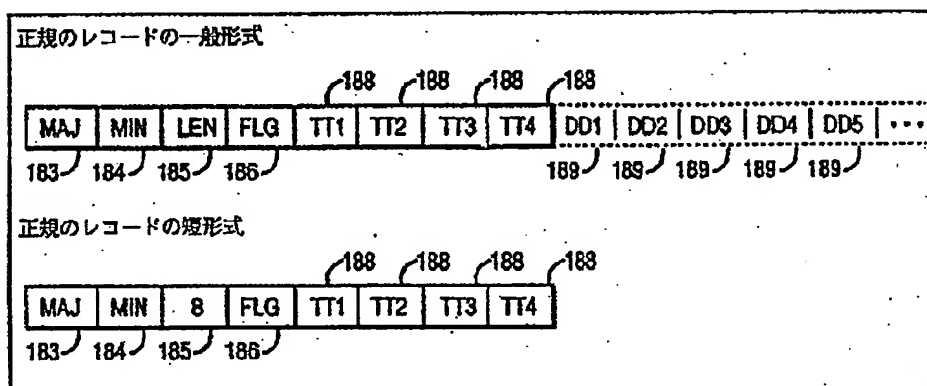
【図7】



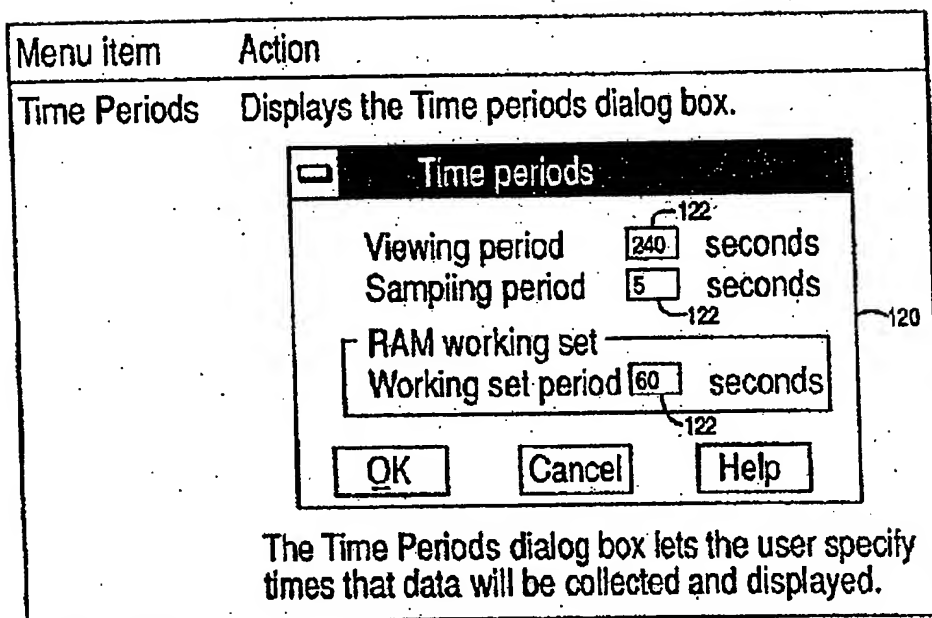
【図6】



【図12】

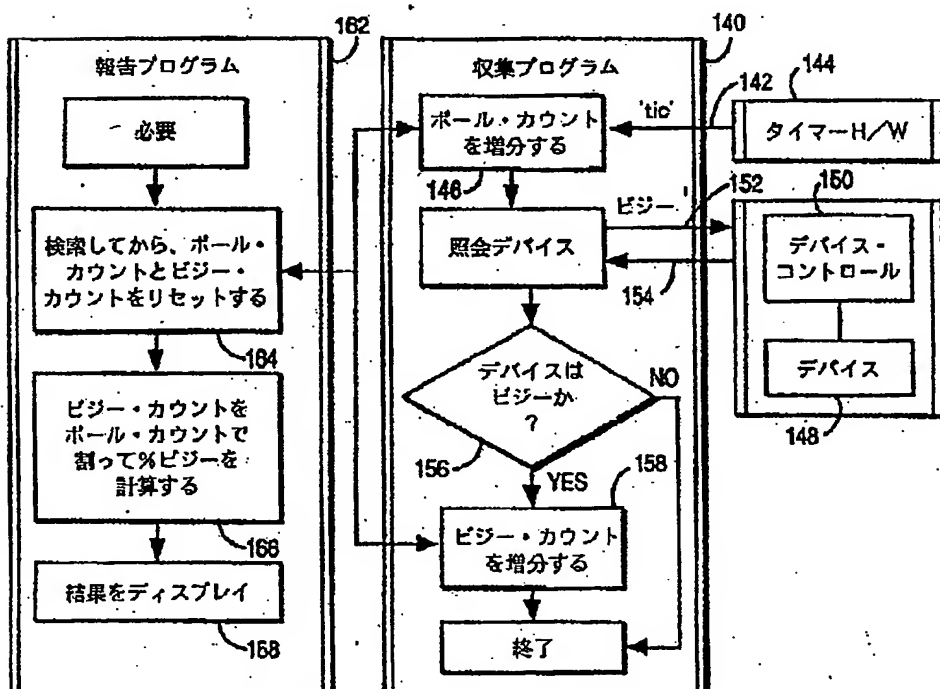


【図8】

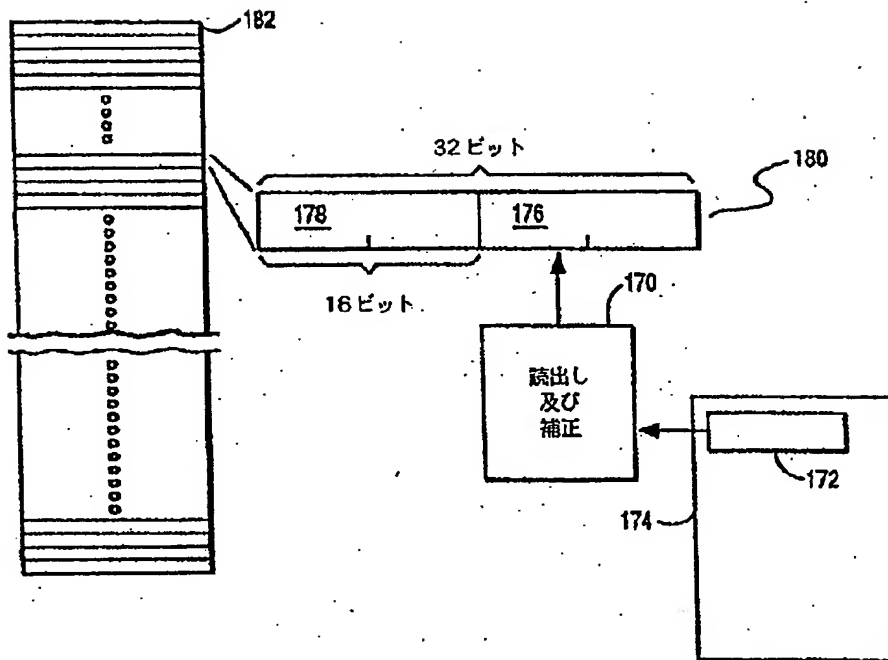


【図10】

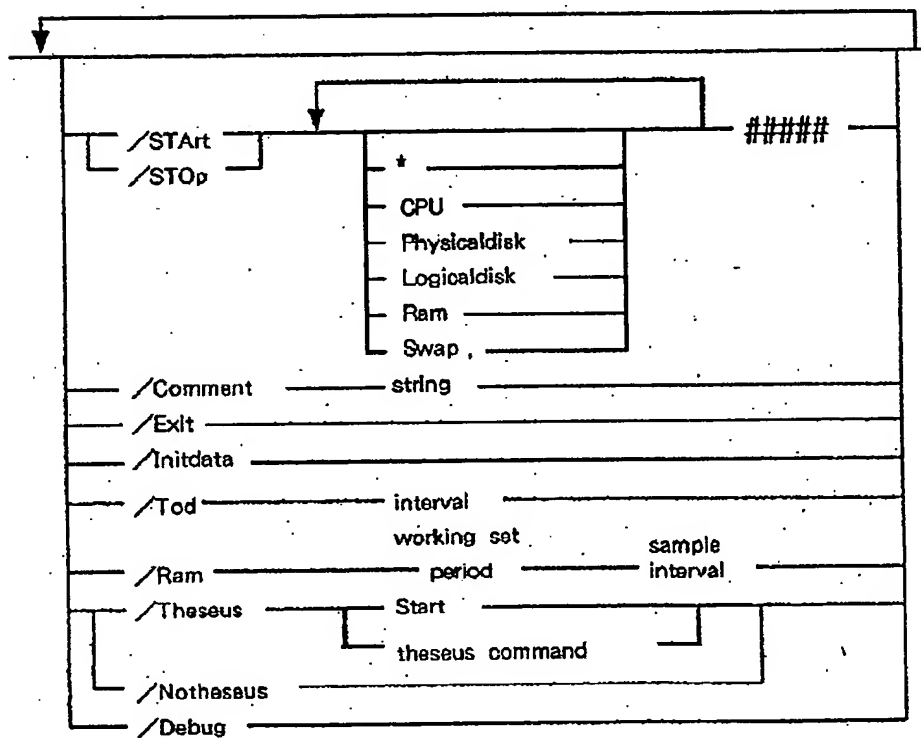
“周辺装置デバイス利用率の測定技術”の図



【図11】



【図14】



【図15】

SPM/2 トレース・パイプ・レコード			
レコード記述 (資源)	トレース・ パイプ・ コード	データ	サイズ
プロセス情報 (CPU)	01	プロセス ID	ワード
		プロセス名	ASCIIZ スtring
プロセス・スイッチ (CPU)	02	プロセス ID (ディスパッチされたプロセス)	ワード
		前のプロセスの実行時間 (タイマー的)	ダブルワード
		前のプロセスの中断時間 (タイマー的)	ダブルワード
1日の時間 (非タイプ)	10	時間	バイト
		分	バイト
		秒	バイト
		確保	1 バイト
		1日の直前のレコードからの時間 (タイマー的)	ダブルワード
		日	バイト
		月	バイト
		年	ワード

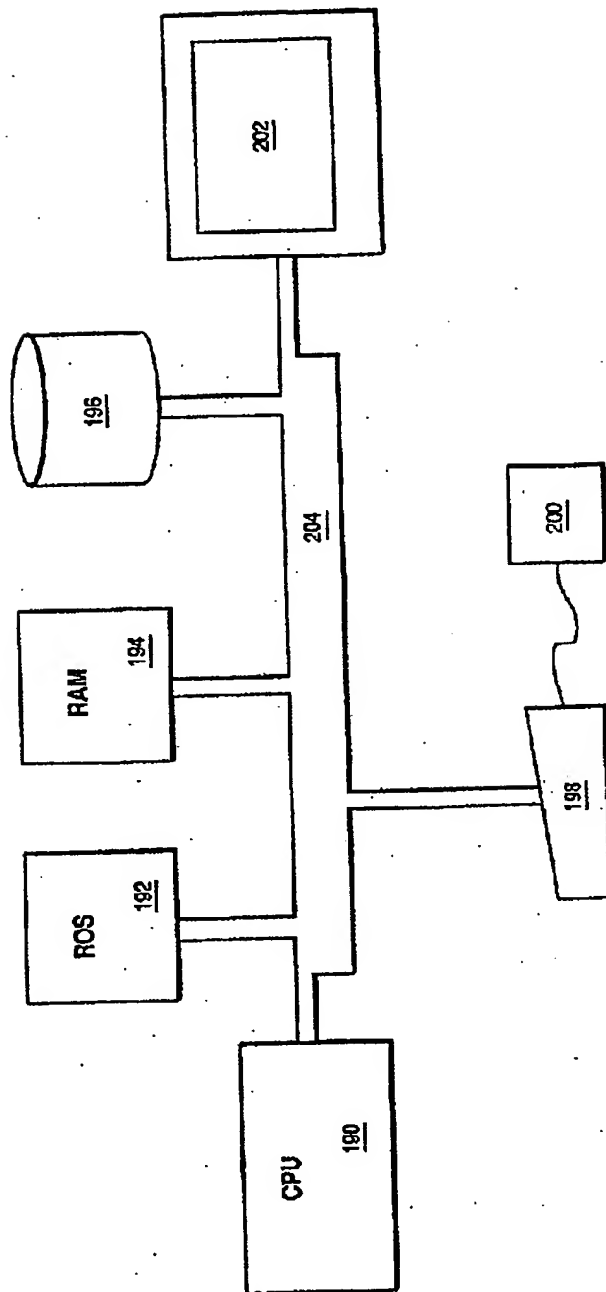
【図16】

SPM/2 トレース・パイプ・レコード			
レコード記述 (資源)	トレース・ パイプ・ コード	データ	サイズ
システム情報 (非タイプ)	12	物理ディスクの数	ワード
		第1物理ディスクのID	ワード
		全装着RAM (バイト)	ダブルワード
		確保	22 バイト
RAM (RAM)	13	全てのスワップ可能/放棄可能のRAM (バイト)	ダブルワード
		ワーキング・セットのRAM (バイト)	ダブルワード
		ワーキング・セット以外のRAM	ダブルワード
		フリー RAM	ダブルワード
		ワーキング・セット期間	ダブルワード
TRACECMD (非タイプ)	17	確保	バイト
データ・ オーバーフロー (非タイプ)	18	確保	4 バイト
ディスク読出し (物理ディスク)	21	物理ディスクID	ワード
		経過時間 (タイマー的)	ダブルワード
		セクタ数	ワード
ディスク書込み (物理ディスク)	22	物理ディスクID	ワード
		経過時間 (タイマー的)	ダブルワード
		セクタ数	ワード

【図17】

SPM/2 トレース・パイプ・レコード			
レコード記述 (資源)	トレース・ パイプ・ コード	データ	サイズ
ディスク書き込み検査 (物理ディスク)	23	物理ディスクID	ワード
		経過時間 (タイマー的)	ダブルワード
		セクタ数	ワード
DOS オープン (論理ディスク)	24	プロセスID	ワード
		ファイル spec	ASCIIZ スtring
		ファイル・ハンドル	ワード
DOS 読出し (論理ディスク)	25	プロセスID	ワード
		ファイル・ハンドル	ワード
		バイト数	ワード
DOS 書き込み (論理ディスク)	27	プロセスID	ワード
		ファイル・ハンドル	ワード
		バイト数	ワード
DOS クローズ (論理ディスク)	28	プロセスID	ワード
		ファイル・ハンドル	ワード
スワップ・イン (スワップ)	31	経過時間 (タイマー的)	ダブルワード
		セグメント長 (バイト)	ダブルワード
スワップ・アウト (スワップ)	32	経過時間 (タイマー的)	ダブルワード
		セグメント長 (バイト)	ダブルワード
注釈 (非タイプ)	40	注釈String (最大サイズ40キャラクタ+空白)	ASCIIZ スtring

【図18】



フロントページの続き

(72)発明者 サミュエル・リー・エムリツク  
アメリカ合衆国テキサス州、オースティン、  
ウエルダン・スプリングス・コート 8003番地

(72)発明者 テイモシー・マンフレッド・ホルク  
アメリカ合衆国テキサス州、オースティン、  
ロメリア 1401番地

(72)発明者 ジェームス・ホイエット・サマーズ  
アメリカ合衆国テキサス州、ラウンド・ロ  
ック、フrint・ロック・ドライブ 2001  
番地